CONVEX AND CONIC OPTIMIZATION
Homework 3

NILAVA METYA
nilava.metya@rutgers.edu
nm8188@princeton.edu

March 21, 2024

Problem 1

Define M_C as the following function of a convex set C in $\mathbb{R}^n$:

$$M_C(x) = \inf\left\{t > 0 \;\middle|\; \frac{x}{t} \in C\right\}$$

over the domain

$$dom(M_C) = \left\{x \in \mathbb{R}^n \;\middle|\; \frac{x}{t} \in C \text{ for some } t > 0\right\}.$$

1. Show that M_C is a convex function.
2. Suppose C is also compact, origin symmetric ($x \in C$ if and only if $-x \in C$), and has nonempty interior. Show that M_C is a norm over $\mathbb{R}^n$. What is its unit ball?
3. Show that an even degree homogeneous polynomial is convex if and only if it is quasiconvex. (Hint: use what you proved in the previous parts of this question.)

Solution

Let's write M for M_C. First we prove an important lemma that will be needed. Using this we will show that $dom(M)$ is indeed convex, so one can actually speak about convexity of M. Denote $E \coloneqq dom(M)$.

We say $D \subseteq \mathbb{R}^n$ is a *strict cone* if $\lambda x \in D \; \forall \lambda > 0, x \in D$.

Proposition 1
If $D \subset \mathbb{R}^n$ is a strict cone and $x + y \in D \; \forall x, y \in D$, then D is convex.

Proof. $x, y \in D, t \in (0,1) \implies tx, (1-t)y \in D \implies tx + (1-t)y \in D$. Trivial for $t \in \{0, 1\}$. ■

Claim 2
E is a strict cone.

Proof. If $x \in E$ then $\frac{x}{t} \in C$ for some $t > 0$ whence $\frac{\lambda x}{\lambda t} \in C \; \forall \lambda > 0$ so $\lambda x \in E \; \forall \lambda > 0$. ■

Claim 3
If $x, y \in E$ then $x + y \in E$.

Proof. $x, y \in E \implies \exists t, s > 0$ such that $\frac{x}{t}, \frac{y}{s} \in C$, whence $\frac{x+y}{s+t} = \frac{t}{s+t} \cdot \frac{x}{t} + \frac{s}{s+t} \cdot \frac{y}{s} \in C$, so $x + y \in E$. ■

This means that E is convex.

1. Say $x, y \in E, \lambda \in [0,1]$. First we note that there are sequences $\{a_n\}_n, \{b_n\}_n$ of positive reals such that $\frac{x}{a_n}, \frac{y}{b_n} \in C\ \forall n$ and $\lim_{n\to\infty} a_n = M(x)$, $\lim_{n\to\infty} b_n = M(y)$. Consider $c_n = \lambda a_n + (1-\lambda)b_n$. By construction, $\{c_n\}$ converges to $\lambda M(x) + (1-\lambda)M(y)$. Then $\frac{\lambda x + (1-\lambda)y}{c_n} = \frac{\lambda}{c_n} \cdot x + \frac{1-\lambda}{c_n} \cdot y = \frac{\lambda a_n}{c_n} \cdot \frac{x}{a_n} + \frac{(1-\lambda)b_n}{c_n} \cdot \frac{y}{b_n} \in C$ because $0 \le \lambda a_n, (1-\lambda)b_n \le c_n$ and $\frac{\lambda a_n}{c_n} + \frac{(1-\lambda)b_n}{c_n} = 1$. This gives the following

$$\begin{aligned} & M(\lambda x + (1-\lambda)y) \le c_n = \lambda a_n + (1-\lambda)b_n \qquad \forall n \in \mathbb{N} \\ \implies & M(\lambda x + (1-\lambda)y) \le \lim_{n\to\infty} c_n = \lambda M(x) + (1-\lambda)M(y) \\ \implies & M \text{ is convex.} \end{aligned}$$

2. We now assume C is also compact, origin symmetric and has nonempty interior.

   The following claims show that $(dom(M) =)E = \mathbb{R}^n$. In the following, $e_1, \cdots, e_n$ are the standard basis vectors of $\mathbb{R}^n$.

   Claim 4
   $\exists r > 0$ such that $\pm re_i \in C$ for each $1 \le i \le n$.

   Proof. Say $p \in C^o$. Here C^o is the interior of C, which is nonempty by assumption. So there is a (closed) ball $B(p,r)$ of radius $r > 0$ such that $B(p,r) \subseteq C^o \subseteq C$. Fix any $1 \le i \le n$. By definition, $p + re_i, p - re_i \in B(p,r) \subseteq C$. By symmetry of C, $-p + re_i = -(p - re_i) \in C$. By convexity of C, $re_i = \frac{(p + re_i) + (-p + re_i)}{2} \in C$. By arbitrariness of i and symmetry of C, $\pm re_i \in C$. ■

   Claim 5
   $E = \mathbb{R}^n$.

   Proof. Pick an arbitrary $q = \sum_{i=1}^{n} a_i e_i \in \mathbb{R}^n$ with $a_i \in \mathbb{R}$. Clearly $\pm e_i \in E\ \forall 1 \le i \le n$ by the previous claim. By symmetry of C, $0 \in C$ whence $0 \in E$. This, along with claim 2, makes E a cone. So $a_i e_i \in E\ \forall i$. By claim 3, $q \in E$. So $E = \mathbb{R}^n$. ■

   Thus M is indeed a well-defined function on $\mathbb{R}^n$. We will now show that it is a norm, that is, we verify the respective conditions:

   - (Positivity)
     By definition, $M(x) \ge 0\ \forall x \in \mathbb{R}^n$. $M(0) = 0$ because $0 \in C$ whence $\frac{0}{t}\ \forall t > 0$.
     Now say $x \in \mathbb{R}^n \setminus \{0\}$. Since C is bounded (because compact), C is contained in some $B(0,r)$ with $r > 0$. This means if $t > 0$ is such that $\frac{x}{t} \in C$ then $0 < \frac{\|x\|}{t} = \left\|\frac{x}{t}\right\| \le r \implies t \ge \frac{\|x\|}{r}$ whence $M(x) \ge \frac{\|x\|}{r} > 0$.
   - (Homogeneity)
     Let $x \in \mathbb{R}^n, a \in \mathbb{R}$. If $a = 0$ then clearly $M(ax) = M(0) = 0 = |a| \cdot M(x)$.
     So assume $a \ne 0$. Then there is a sequence $\{t_n\}$ of positive reals such that $\frac{x}{t_n} \in C$ and $\lim t_n = M(x)$. By symmetry of C, $\frac{ax}{|a|t_n} = \frac{a}{|a|} \cdot \frac{x}{t_n} \in C$. But $|a|\, t_n > 0\ \forall n$ and $\lim(|a|\, t_n) = |a|\, M(x)$. This shows that $M(ax) \le |a|\, M(x)$. a was any nonzero real, so we also have $M(\frac{1}{a} \cdot (ax)) \le \frac{1}{|a|} M(ax)$. Combining the two we get $M(ax) \le |a|\, M(x) = |a|\, M(\frac{1}{a} \cdot (ax)) \le M(ax)$ whence $M(ax) = |a|\, M(x)$.
   - (Triangle inequality)

Let $x, y \in \mathbb{R}^n$. We already showed that M is convex. Thus

$$M(x+y) = M\left(\frac{2x+2y}{2}\right) \overset{\text{convexity}}{\leq} \frac{M(2x)+M(2y)}{2} \overset{\text{homogeneity}}{=} M(x)+M(y).$$

Now we try to find the unit ball of M, that is, $B_M \coloneqq \{x \in \mathbb{R}^n \mid M(x) \leq 1\}$. It trivially contains 0. Note that if $x \in C$ then $\frac{x}{1} \in C$ whence $M(x) \leq 1$, proving that $C \subseteq B_M$. Suppose $x \in \mathbb{R}^n \setminus \{0\}$ is such that $M(x) \leq 1$. Then there is a sequence of positive reals $\{a_n\}_n$ such that $\frac{x}{a_n} \in C$ and $\lim a_n = M(x) \leq 1$. Then $\frac{x}{M(x)} = \lim \frac{x}{a_n} \in C$ because C is closed. $(1 - \frac{1}{M(x)}) \cdot 0 + \frac{1}{M(x)} \cdot x \in C$ because C is convex and $\frac{1}{M(x)} \in (0, 1]$. So the unit ball is C itself.

3. Let f be a homogeneous even-degree polynomial on $\mathbb{R}^n$ of degree $2d$. Let $C = \{x \in \mathbb{R}^n \mid f(x) \leq 1\}$.

   Assume f is quasiconvex. Then C is convex because C is a sublevel set of f. Consider $M(x) = \inf\left\{t > 0 \;\middle|\; \frac{x}{t} \in C\right\}$. It is clear that $f\left(\frac{x}{\max(f(x),1)^{\frac{1}{2d}}}\right) = \frac{f(x)}{\max(f(x),1)} \leq 1$ so that $\frac{x}{t} \in C$ with $t = \max(f(x),1)^{\frac{1}{2d}} > 0$ for every x whence $dom(M) = \mathbb{R}^n$. We used the fact that $f(ax) = a^{2d} f(x)$ by homogeneity of f. Next we note that $f \geq 0$ because if there were some $a \in \mathbb{R}^n$ such that $f(a) = f(-a) < 0$ then by convexity of $\{x \in \mathbb{R}^n \mid f(x) \leq f(a) (< 0)\}$, it would have to contain 0 because it contains $\pm a$; but it cannot contain 0 since $f(0) = 0 \not< 0$. Observe the following for $t > 0$ and fixed (arbitrary) $x \in dom(M) = \mathbb{R}^n$:

$$\frac{x}{t} \in C \iff f\left(\frac{x}{t}\right) \leq 1 \iff \frac{f(x)}{t^{2d}} \leq 1 \iff f(x) \leq t^{2d}.$$

   This means that $M(x) = f(x)^{\frac{1}{2d}}$. M is convex because of the earlier problem. So $f(x) = M(x)^{2d}$ is a positive power of a non-negative convex function, namely M, and thus convex.

Problem 2

Let A be an integral matrix. Show that A is totally unimodular if and only if for every integral vector b, the polyhedron $\{x \mid x \geq 0, Ax \leq b\}$ is integral. (Hint: If A is not totally unimodular, use the inverse of a submatrix which does not have determinant $\{0, -1, +1\}$ to construct an integer vector b that generates a non-integral vertex in the polyhedron.)

Solution

We already saw that if A is TUM then the given polytope is integral for every integral b.

Now assume $A_{m\times n}$ is not TUM. So there is a $k \times k$ submatrix whose determinant is neither of $0, \pm 1$. WLOG, say the matrix formed by first k rows and first k columns of A have determinant other than $0, \pm 1$ (otherwise simply rearrange the rows and columns and the TUM-ness will not be violated). Call it D. The given polytope can be described with a single matrix, namely $B = \begin{bmatrix} A \\ -\mathbf{1}_{n\times n} \end{bmatrix}$ where $\mathbf{1}_{n\times n}$ is the $n \times n$ identity matrix. Spelt out, the given polytope is same as $\{x \in \mathbb{R}^n \mid Bx \leq d\}$ where $d = \begin{bmatrix} b^\top & \mathbf{0}_{1\times n}^\top \end{bmatrix}$. So B looks like

$$B = \begin{bmatrix} D & D' \\ D'' & D''' \\ -\mathbf{1}_{k\times k} & \mathbf{0} \\ \mathbf{0} & -\mathbf{1}_{(n-k)\times(n-k)} \end{bmatrix}$$

where $\det D \notin \{0, \pm 1\}$. In particular D is invertible.

Note that $\det(DD^{-1}) = 1$ whence one of $\det D$ or $\det D^{-1}$ must be non-integral. But D has integer entries, consequently its determinant is an integer. This means that $\det D^{-1} \notin \mathbb{Z}$. So D^{-1} has a non-integer entry, say at (s, t). let $e_t \in \mathbb{R}^k$ be the t^{th} basis vector. Note $D^{-1}e_t \notin \mathbb{Z}^k$. Choose $y \in \mathbb{Z}^k$ such that $D^{-1}e_t + y > 0$ and consider the vector

$$x = \begin{bmatrix} D^{-1}e_t + y \\ \mathbf{0}_{(n-k)\times 1} \end{bmatrix}.$$

Take b to be the vector in $\mathbb{R}^m$ which looks like

$$b = \begin{bmatrix} (e_t + Dy)_{k\times 1} \\ \\ \lceil D''D^{-1}e_t \rceil + D''y + \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{(m-k)\times 1} \end{bmatrix} \in \mathbb{R}^m$$

where $\lceil \cdot \rceil$ is taken coordinate-wise. In fact, $b \in \mathbb{Z}^m$ because D, y have integer entries. So d is just this with n concatenated 0's (as defined earlier). Notice that x satisfies

- $\begin{bmatrix} D & D' \end{bmatrix} x = e_t + Dy = b_{1:k} = d_{1:k}$.
- $\begin{bmatrix} D'' & D''' \end{bmatrix} x = D''D^{-1}e_t + D''y < b_{k+1:m} = d_{k+1:m}$.
- $\begin{bmatrix} -\mathbf{1}_{k\times k} & \mathbf{0} \end{bmatrix} x = D^{-1}e_t + y < 0 = d_{m+1:m+k}$.
- $\begin{bmatrix} 0 & -\mathbf{1}_{(n-k)\times(n-k)} \end{bmatrix} x = 0 = d_{m+k+1:m+n}$.

Here $v_{i:j}$ represents the sub-vector of v formed by taking the coordinates $i, i+1, \cdots, j$. Note that the matrix $\begin{bmatrix} D & D' \\ 0 & -\mathbf{1}_{(n-k)\times(n-k)} \end{bmatrix}$ has determinant $= \det D \cdot (-1)^k \neq 0$ whence these rows are linearly independent. In short, we showed that x satisfies all linear constraints and is tight at exactly n linearly independent constraints (given by the first and fourth bullet points above). However, x is not integral because $D^{-1}e_t \notin \mathbb{Z}^k$ by construction. This x is a non-integral vertex of the given polytope even though $b \in \mathbb{Z}^m$.

Problem 3

Solution

1. We are given the data $B^{\max}, A, \mathcal{T}, \mathcal{D}^{\text{target}}, \mathcal{D}^{\text{other}}$. The optimization variable is b. Thus the problem we want to formulate is

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

   Note that the above constraints are all linear, but not the objective function. This can be made linear (with additional linear constraints) by bringing extra variables. That is, introduce variables u_i for each $i \notin \mathcal{T}$ and require that $u_i \geq d_i - D^{\text{other}} \ \forall i$ and $u_i \geq 0 \ \forall i$. So the final linear program becomes

$$\begin{aligned} \min \ & \sum_{i \notin \mathcal{T}} u_i \\ \text{s.t. } & d = Ab \\ & d_i \geq D^{\text{target}} \ \forall i \in \mathcal{T} \\ & 0 \leq b_i \leq B^{\max} \ \forall 1 \leq i \leq n \\ & 0 \leq u_i \ \forall 1 \leq i \leq m, i \notin \mathcal{T} \\ & d_i - D^{\text{other}} \leq u_i \ \forall 1 \leq i \leq m, i \notin \mathcal{T}. \end{aligned} \tag{2}$$

   This is clearly a linear program with linear objective and linear inequalities and equalities.

2. See next page for code and histogram.

   Observation from the histogram: Since the problem had *required* the tumor radiation to be ≥ 1 and *tried to push* the other radiation to be ≤ 0.25, there is a big gap in the histogram from around 0.30 to 1.00 where there is no radiation level. And there's quite many radiation levels around 0.25 which is explained by the peak in the graph around 0.25. Also quite a few beams around 1 on the tumor cells.

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

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

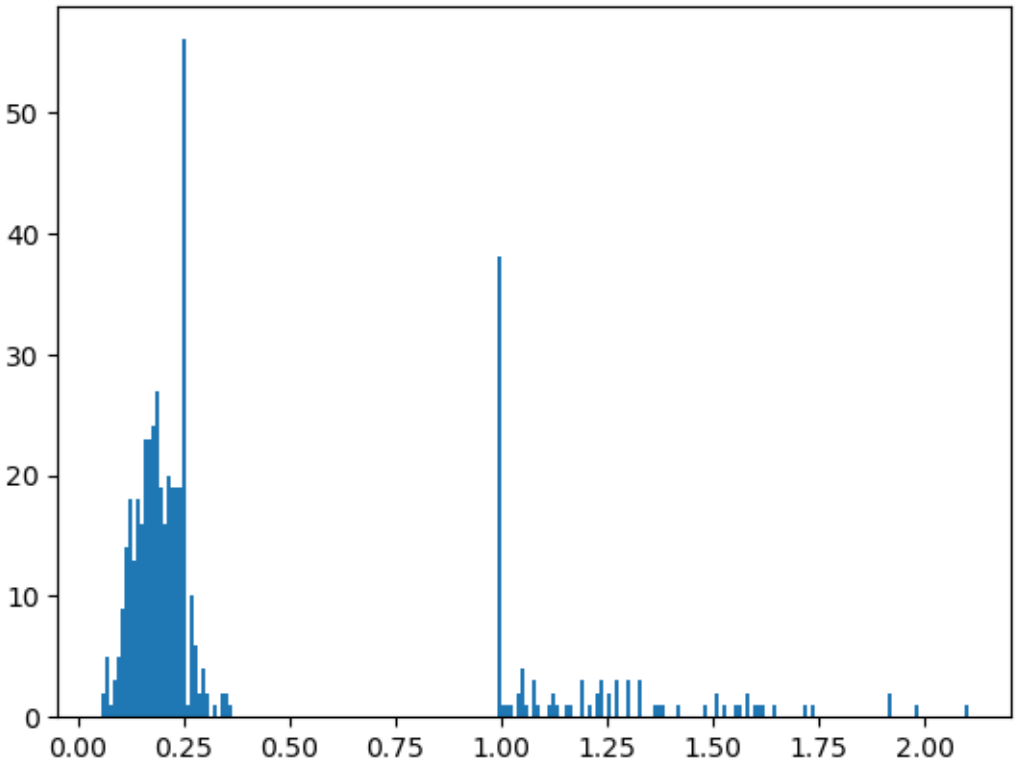
[4]:
```
dtumor = cp.Variable(tumor, 'dtumor')
dother = cp.Variable(other, 'dother')
b = cp.Variable(n, 'b')
f = cp.maximum(dother - Dother, 0)
obj = cp.sum(f)
```

```
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]:
```
#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

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

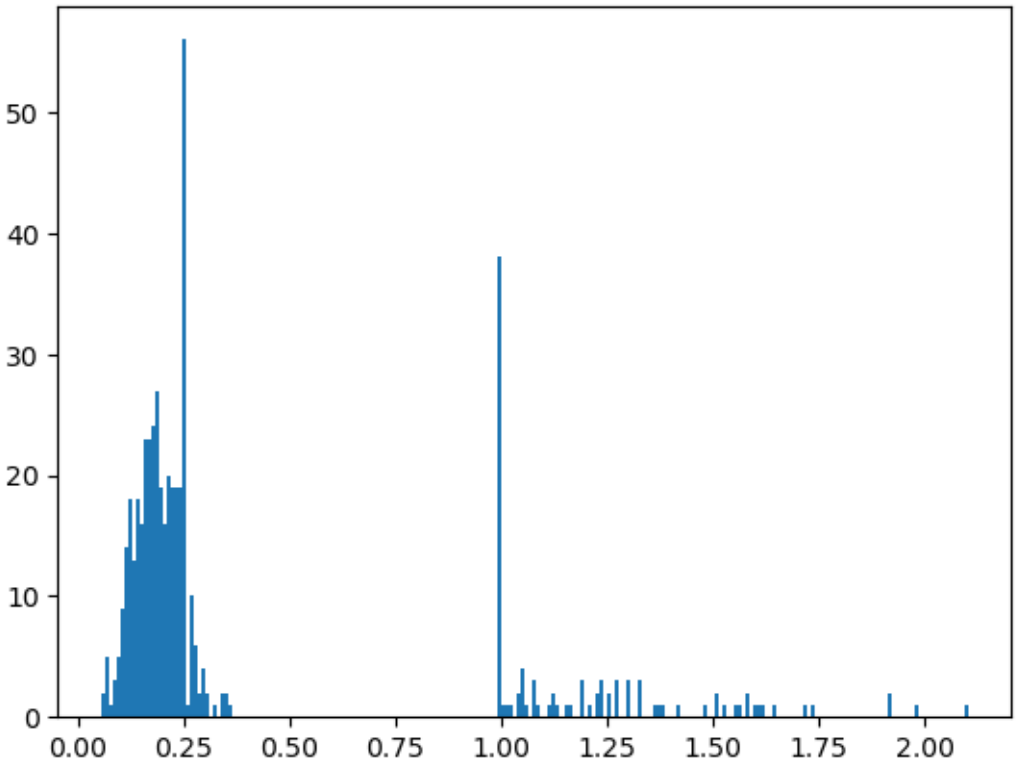
[6]:
```
u = cp.Variable(other, 'u')
dtumor = cp.Variable(tumor, 'dtumor')
dother = cp.Variable(other, 'dother')
b = cp.Variable(n, 'b')
f = cp.maximum(dother - Dother, 0)
obj = cp.sum(f)
cons = [Aother @ b == dother, Atumor @ b == dtumor, dtumor >= Dtarget, b >= 0,␣
 ↪Bmax >= b, u >= 0, u >= dother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

Problem 4

Recall our Support Vector Machines application of convex optimization from lecture. We have m feature vectors $x_1, \cdots, x_m \in \mathbb{R}^n$ with each x_i having a label $y_i \in \{-1, 1\}$. The goal is to find a linear classifier, that is a hyperplane $a^\top x - b$, where $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$, by solving the optimization problem

$$\begin{aligned} &\min_{a,b} \ \|a\| \\ &\text{s.t.} \ \ y_i(a^\top x_i - b) \geq 1 \ \forall i = 1, \cdots, m. \end{aligned} \tag{3}$$

We will then use this classifier to classify new data points.

1. Prove that the solution to 3 is unique.
2. Show that 3 is equivalent to

$$\begin{aligned} &\max_{a,b,t} \ t \\ &\text{s.t.} \ \ y_i(a^\top x_i - b) \geq t \ \forall i = 1, \cdots, m \\ &\qquad \|a\| \leq 1, \end{aligned} \tag{4}$$

   which is easier to interpret in terms of finding a classifier with maximum margin. Show that if 3 is feasible (with a positive optimal value), then 4 is feasible (and has a positive optimal value). Conversely, show that if 4 is feasible (with a positive optimal value), then 3 is feasible (and has a positive optimal value).
3. Assume the optimal value of 4 is positive. Show that an optimal solution of 4 always satisfies $\|a\| = 1$.
4. Prove that the Euclidean distance of a point $v \in \mathbb{R}^n$ to a hyperplane $a^\top z = b$ is given by $\dfrac{\left|a^\top v - b\right|}{\|a\|}$.

Solution

1. We'll prove that if a solution exists (in particular, the data is linearly separable), it must be unique.

   Say $(a, b), (u, v) \in \mathbb{R}^n \times \mathbb{R}$ are distinct solutions to 3. So, $\|a\| = \|u\|$ and $y_i(u^\top x_i - v) \geq 1, y_i(a^\top x_i - b) \geq 1 \ \forall i$. Suppose $a \neq u$. Then $\left\|\dfrac{a+u}{2}\right\|^2 < \dfrac{\|a\|^2 + \|u\|^2}{2} = \|a\|^2$ because $\|\cdot\|^2$ is strictly convex (Hessian has all eigenvalues $1 > 0$). It follows that 3 takes a strictly lower value at $\frac{1}{2}(a, b) + \frac{1}{2}(u, v)$, namely $\left\|\frac{a+u}{2}\right\|$, than at (a, b) and (u, v). This contradicts the optimality of $(a, b), (u, v)$.
   This forces $a = u$ and $b \neq v$.

   Claim 6
   There must exist indices $s \neq t$ such that $y_s = -y_t = +1$ and $a^\top x_s + b = +1, a^\top x_t + b = -1$.

   Proof. We will use the fact that (a, b) is a minimizer. Suppose $a^\top x_i - b > 1 \forall i \in P \coloneqq \{j \in \mathbb{N} | y_j > 0, 1 \leq j \leq m\}$. Take $m = \min_{i \in P} a^\top x_i - b - 1 > 0$. Note that $\left(\dfrac{a}{1 + m/2}, \dfrac{b + m/2}{1 + m/2}\right)$ is feasible and has lower objective value. It is feasible because if $i \notin P$ then $y_i \left(\dfrac{a^\top x_i - b - m/2}{1 + m/2}\right) = \dfrac{y_i(a^\top x_i - b) + m/2}{1 + m/2} \geq \dfrac{1 + m/2}{1 + m/2} = 1$ and if $i \in P$ then $y_i \left(\dfrac{a^\top x_i - b - m/2}{1 + m/2}\right) = \dfrac{1 + m - m/2}{1 + m/2} = 1$. Now clearly $m > 0$ so that the optimal value of this new feasible point, namely $\dfrac{\|a\|}{1 + m/2}$, is strictly less than the value of the assumed minimizer, a contradiction. So there is some s such that $a^\top x_s - b = 1$. A similar argument gives existence of t by simply swapping the roles of ± 1 in the above. ■

Let s, t be as above. Recall $a = u$. So $a^\top x_s - v \geq 1 = a^\top x_s - b \implies b - v \geq 0$. And $a^\top x_t - v \leq -1 = a^\top x_t - b \implies b - v \leq 0$. These two together imply that $b = v$.

2. Say 3 is feasible with positive optimal value and optimal solution (a, b). Let $\tau \coloneqq \min\limits_{1 \leq i \leq m} \dfrac{\left|a^\top x_i - b\right|}{\|a\|}$. We will show that $(\alpha, \beta, t) = \left(\dfrac{a}{\|a\|}, \dfrac{b}{\|a\|}, \tau\right)$ is an optimal solution to 4 with optimal value τ.

   Feasibility: $y_i\left(\alpha^\top x_i - \beta\right) = \dfrac{y_i\left(a^\top x_i - b\right)}{\|a\|} = \dfrac{\left|a^\top x_i - b\right|}{\|a\|} \overset{\text{by definition}}{\geq} \tau \; \forall i$. Furthermore, $\|\alpha\| = \frac{\|a\|}{\|a\|} = 1$.

   Optimality: Note that because 3 has an optimal solution, there is some separating hyperplane that strictly separates the points having label $+1$ against the points having label -1, because convex hull of both of these sets are compact. Let (c, d, t) be feasible to 4 such that $t > 0$. Here $c \neq 0$ because otherwise $-y_i b \geq t \geq 0 \forall i$ which is impossible because there exist both $+1, -1$ labels. So, $y_i(c^\top x_i - d) \geq t \; \forall i$ and $\|c\| \leq 1$. Consider $t' = \min\limits_{1 \leq i \leq m} y_i(c^\top x_i - d) = \min\limits_{1 \leq i \leq m} \left|c^\top x_i - d\right|$. This is the margin for the chosen hyperplane. Trivially it satisfies that $t' \geq t$. Note that $t' \geq y_i(c^\top x_i - d) \; \forall i \implies 1 \geq y_i\left(\frac{c^\top}{t'} x_i - \frac{d}{t'}\right) \; \forall i$. This means $\left(\frac{c}{t'}, \frac{d}{t'}\right)$ is feasible to 3 whence $\|a\| \leq \frac{\|c\|}{t'} \leq \frac{1}{t'} \implies t' \leq \frac{1}{\|a\|}$. Now note that $y_i\left(\alpha^\top x_i - \beta\right) = \frac{y_i\left(a^\top x_i - b\right)}{\|a\|} \geq t' y_i\left(a^\top x_i - b\right) \geq t' \geq t \forall i$ where the second-last inequality is true because $y_i\left(a^\top x_i - b\right) \geq 1$ due to the feasibility of (a, b) to 3. It follows that $\tau \geq t$ by taking min over all $1 \leq i \leq m$.

   Say 4 is feasible with positive optimal value and has optimal solution (a, b, t). We will show that 3 has optimal solution $(\alpha, \beta) = \left(\frac{a}{t}, \frac{b}{t}\right)$.

   Feasibility: By feasibility of (a, b, t) to 4, we can guarantee that $y_i(\alpha^\top x_i - \beta) = \frac{y_i(a^\top x_i - b)}{t} \geq \frac{t}{t} = 1 \forall i$.
   Optimality: By optimality of (a, b, t) to 4, it must happen that one of the inequalities $t \leq y_i(a^\top x_i - b)$ is an equality, otherwise t can be improved (increased) to $t^* = \min\limits_{1 \leq i \leq n} y_i(a^\top x_i - b) > t$. So $t = \min\limits_{1 \leq i \leq n} y_i(a^\top x_i - b)$ and say this equality happens at $i = i_1$. Let (c, d) be feasible to 3. So $y_i(c^\top x_i - d) \geq 1 \; \forall i$. $c \neq 0$ because some label is $+1$ and some label is -1. So $y_i(c^\top x_i - d) \geq 1 \; \forall i$. Consider $\gamma = \dfrac{c}{\|c\|}, \delta = \dfrac{d}{\|c\|}, s = \min\limits_{1 \leq i \leq m} y_i(\gamma^\top x_i - \delta)$. Say, this min occurs at $i = i_0$. Note that (γ, δ, s) is feasible to 4 because $\|\gamma\| = 1$ and $s \leq y_i(\gamma^\top x_i - \delta) \; \forall i$ by definition. So $t \geq s = y_{i_0}(\gamma^\top x_{i_0} - \delta) = \dfrac{y_{i_0}(c^\top x_{i_0} - d)}{\|c\|} \geq \dfrac{1}{\|c\|}$ where the last ineuqality is true because (c, d) is feasible to 3. It thus follows that $\|c\| \geq \frac{1}{t} \geq \frac{\|a\|}{t} = \|\alpha\|$. So $\|\alpha\|$ is at most any value that 3 can obtain, meaning that (α, β) is optimal.

3. Say (a, b, t) is an optimal solution to 4 such that the optimal value $t > 0$. For the sake of contradiction, assume $\|a\| < 1$ (because $\|a\| \leq 1$ is guaranteed). Consider $(\alpha, \beta, \tau) = \left(\frac{a}{\|a\|}, \frac{b}{\|a\|}, \frac{t}{\|a\|}\right)$. Then $y_i(a^\top x_i - b) \geq t \; \forall i \implies y_i(\alpha^\top x_i - \beta) \geq \tau \; \forall i$ by multiplying by $\frac{1}{\alpha} > 0$ throughout; and $\|\alpha\| = 1 \leq 1$. So (α, β, τ) is feasible to 4. However, $\tau > t$ is a strictly better (more) optimal value, a contradiction. This forces $\|a\| = 1$.

4. We want to solve the optimization problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \; & \|v - x\| \\ \text{s.t. } & a^\top x = b. \end{aligned}$$

   The constraint here is linear, hence the constraint set Ω is convex. We will show that $f = v + \dfrac{b - a^\top v}{\|a\|^2} a$ is an optimal solution. Note that this satisfies $a^\top f = a^\top v + \frac{b - a^\top v}{\|a\|^2} \|a\|^2 = b$, i.e., f is feasible.

Let y be any point on the hyperplane, that is, $a^\top y = b$. Then

$$\begin{aligned}
\|v-y\|^2 &= \|v-f+f-y\|^2 \\
&= \|v-f+f-y\|^2 \\
&= \|v-f\|^2 + \|f-y\|^2 + 2(v-f)^\top (f-y) \\
&\geq \|v-f\|^2 + 2(v-f)^\top (f-y) \\
&= \|v-f\|^2 + \left(\frac{a^\top v - b}{\|a\|^2}\right) a^\top (f-y) \\
&= \|v-f\|^2 + \left(\frac{a^\top v - b}{\|a\|^2}\right) (b-b) \\
&= \|v-f\|^2
\end{aligned}$$

This proves that f is an optimal solution with optimal value $\|v-f\| = \left\| \frac{b-a^\top v}{\|a\|^2} a \right\| = \frac{|b-a^\top v|}{\|a\|}$.

Problem 5

Solution

1. See code next page. The values of optimal a, b for each γ are tabulated as follows.

γ	Optimal a	Optimal b
0.1	$(0.14105247, 0.18277618, -0.73224986, -0.10977297, 0.38083898)$	-3.14700164
1	$(0.20864823, -0.97870147, -1.62007281, -0.4604091, 3.76855067)$	-9.24105061
10	$(0.15062914, -0.91314802, -1.52389243, -0.4642144, 4.82133807)$	-8.80471718

   We assume that the features occur in the order as mentioned in the question: mean income, percentage of hispanics, percentage of whites, percentage of residents with a Bachelor's degree or higher, and population density.

2. `gamma1`, which is 0.1, gives the best prediction – only one wrong prediction, as compared to two for `gamma2` and `gamma3`. In this case, the optimal $a^* = (0.14105247, 0.18277618, -0.73224986, -0.10977297, 0.38083898)$. The positive weights are for the parameters corresponding to mean income, percentage of Hispanics and population density. People who live in very high Hispanic-populated areas or areas of quite high mean income or overall high population density are more likely to vote for Hillary. On the other hand, people living in areas with somewhat high white population (or more population with a bachelors degree or higher) are very likely to vote for Bernie. Here, the weight for the white population significantly dominates all others (in terms of absolute value).

Hillary vs. Bernie

[1]:
```
import cvxpy as cp
import numpy as np
import scipy
mat = scipy.io.loadmat('Hillary_vs_Bernie.mat')
X = mat['features_train']
y = mat['labels_train']
m,n = X.shape
Y = np.zeros((m,m),float)
for i in range(m):
    Y[i][i] = y[i][0]
```

Fitting the model for $\gamma \in \{0.1, 1, 10\}$

The optimization problem

$$\begin{aligned} \min_{a,b,\eta} \; & ||a|| + \gamma \, ||\eta||_1 \\ \text{s.t. } & y_i(a^\top x_i - b) \geq 1 - \eta_i \qquad \forall i = 1, \cdots, m \\ & \eta \geq 0 \end{aligned}$$

to build a linear classifier. Corresponding to the three cases `gamma1 = 0.1`, `gamma2 = 1`, `gamma3 = 10`, the optimal solutions are labelled as `(a1,b1,eta1)`, `(a2,b2,eta2)`, `(a3,b3,eta3)` respectively.

Here's how we deal with the linear separator on the given data. I formed a matrix $(Y_{\text{train}} =)Y = diag(y_1, \cdots, y_m)$. The rows of $(X_{\text{train}} =)X$ are the vectors $x_i^\top$. So $Xa - b\mathbf{1}$ already gives the evaluation of the linear form on these data points $\{x_i\}$. We want to weigh each $x_i^\top a - b$ with y_i: this is achieved by taking $Y(Xa - b\mathbf{1})$ which gives a vector with i^{th} entry being $y_i(x_i^\top a - b)$.

[2]:
```
gamma1 = 0.1
a1 = cp.Variable(n, 'a1')
b1 = cp.Variable(1,'b1')
eta1 = cp.Variable(m, 'eta1')
obj1 = cp.norm(a1) + gamma1 * (cp.norm(eta1,1))
cons1 = [Y@(X@a1-b1) + eta1 >= 1, eta1 >= 0]
problem1 = cp.Problem(cp.Minimize(obj1), cons1)
print(problem1.solve(verbose = True, solver = cp.ECOS))
print("\nOptimal a: ", a1.value, "\nOptimal b:", b1.value)
```

```
===============================================================================
                                     CVXPY
```

```
                                     v1.4.2
============================================================================
(CVXPY) Mar 20 09:52:49 AM: Your problem has 181 variables, 2 constraints, and 0
parameters.
(CVXPY) Mar 20 09:52:49 AM: It is compliant with the following grammars: DCP,
DQCP
(CVXPY) Mar 20 09:52:49 AM: (If you need to solve this problem multiple times,
but with different data, consider using parameters.)
(CVXPY) Mar 20 09:52:49 AM: CVXPY will first compile your problem; then, it will
invoke a numerical solver to obtain a solution.
(CVXPY) Mar 20 09:52:49 AM: Your problem is compiled with the CPP
canonicalization backend.
----------------------------------------------------------------------------
                                  Compilation
----------------------------------------------------------------------------
(CVXPY) Mar 20 09:52:49 AM: Compiling problem (target solver=ECOS).
(CVXPY) Mar 20 09:52:49 AM: Reduction chain: Dcp2Cone -> CvxAttr2Constr ->
ConeMatrixStuffing -> ECOS
(CVXPY) Mar 20 09:52:49 AM: Applying reduction Dcp2Cone
(CVXPY) Mar 20 09:52:49 AM: Applying reduction CvxAttr2Constr
(CVXPY) Mar 20 09:52:49 AM: Applying reduction ConeMatrixStuffing
(CVXPY) Mar 20 09:52:49 AM: Applying reduction ECOS
(CVXPY) Mar 20 09:52:49 AM: Finished problem compilation (took 1.305e-02
seconds).
----------------------------------------------------------------------------
                                Numerical solver
----------------------------------------------------------------------------
(CVXPY) Mar 20 09:52:49 AM: Invoking solver ECOS  to obtain a solution.
----------------------------------------------------------------------------
                                    Summary
----------------------------------------------------------------------------
(CVXPY) Mar 20 09:52:49 AM: Problem status: optimal
(CVXPY) Mar 20 09:52:49 AM: Optimal value: 1.057e+01
(CVXPY) Mar 20 09:52:49 AM: Compilation took 1.305e-02 seconds
(CVXPY) Mar 20 09:52:49 AM: Solver (including time spent in interface) took
4.034e-03 seconds
10.57269665702442

Optimal a:  [ 0.14105247  0.18277618 -0.73224986 -0.10977297  0.38083898]
Optimal b: [-3.14700164]
```

[3]:
```
gamma2 = 1
a2 = cp.Variable(n, 'a2')
b2 = cp.Variable(1,'b2')
eta2 = cp.Variable(m, 'eta2')
obj2 = cp.norm(a2) + gamma2 * (cp.norm(eta2,1))
cons2 = [Y@(X@a2-b2) + eta2 >= 1, eta2 >= 0]
```

```
problem2 = cp.Problem(cp.Minimize(obj2), cons2)
print(problem2.solve(verbose = True, solver = cp.ECOS))
print("\nOptimal a: ", a2.value, "\nOptimal b:", b2.value)
```

```
===============================================================================
                                     CVXPY
                                     v1.4.2
===============================================================================
(CVXPY) Mar 20 09:52:49 AM: Your problem has 181 variables, 2 constraints, and 0
parameters.
(CVXPY) Mar 20 09:52:49 AM: It is compliant with the following grammars: DCP,
DQCP
(CVXPY) Mar 20 09:52:49 AM: (If you need to solve this problem multiple times,
but with different data, consider using parameters.)
(CVXPY) Mar 20 09:52:49 AM: CVXPY will first compile your problem; then, it will
invoke a numerical solver to obtain a solution.
(CVXPY) Mar 20 09:52:49 AM: Your problem is compiled with the CPP
canonicalization backend.
-------------------------------------------------------------------------------
                                  Compilation
-------------------------------------------------------------------------------
(CVXPY) Mar 20 09:52:49 AM: Compiling problem (target solver=ECOS).
(CVXPY) Mar 20 09:52:49 AM: Reduction chain: Dcp2Cone -> CvxAttr2Constr ->
ConeMatrixStuffing -> ECOS
(CVXPY) Mar 20 09:52:49 AM: Applying reduction Dcp2Cone
(CVXPY) Mar 20 09:52:49 AM: Applying reduction CvxAttr2Constr
(CVXPY) Mar 20 09:52:49 AM: Applying reduction ConeMatrixStuffing
(CVXPY) Mar 20 09:52:49 AM: Applying reduction ECOS
(CVXPY) Mar 20 09:52:49 AM: Finished problem compilation (took 9.276e-03
seconds).
-------------------------------------------------------------------------------
                                Numerical solver
-------------------------------------------------------------------------------
(CVXPY) Mar 20 09:52:49 AM: Invoking solver ECOS  to obtain a solution.
-------------------------------------------------------------------------------
                                    Summary
-------------------------------------------------------------------------------
(CVXPY) Mar 20 09:52:49 AM: Problem status: optimal
(CVXPY) Mar 20 09:52:49 AM: Optimal value: 8.944e+01
(CVXPY) Mar 20 09:52:49 AM: Compilation took 9.276e-03 seconds
(CVXPY) Mar 20 09:52:49 AM: Solver (including time spent in interface) took
3.277e-03 seconds
89.43653660717314

Optimal a:  [ 0.20864823 -0.97870147 -1.62007281 -0.4604091   3.76855067]
Optimal b: [-9.24105061]
```

[4]:
```
gamma3 = 10
a3 = cp.Variable(n, 'a3')
b3 = cp.Variable(1,'b3')
eta3 = cp.Variable(m, 'eta3')
obj3 = cp.norm(a3) + gamma3 * (cp.norm(eta3,1))
cons3 = [Y@(X@a3-b3) + eta3 >= 1, eta3 >= 0]
problem3 = cp.Problem(cp.Minimize(obj3), cons3)
print(problem3.solve(verbose = True, solver = cp.ECOS))
print("\nOptimal a: ", a3.value, "\nOptimal b:", b3.value)
```

```
===============================================================================
                                     CVXPY
                                     v1.4.2
===============================================================================
(CVXPY) Mar 20 09:52:49 AM: Your problem has 181 variables, 2 constraints, and 0
parameters.
(CVXPY) Mar 20 09:52:49 AM: It is compliant with the following grammars: DCP,
DQCP
(CVXPY) Mar 20 09:52:49 AM: (If you need to solve this problem multiple times,
but with different data, consider using parameters.)
(CVXPY) Mar 20 09:52:49 AM: CVXPY will first compile your problem; then, it will
invoke a numerical solver to obtain a solution.
(CVXPY) Mar 20 09:52:49 AM: Your problem is compiled with the CPP
canonicalization backend.
-------------------------------------------------------------------------------
                                  Compilation
-------------------------------------------------------------------------------
(CVXPY) Mar 20 09:52:49 AM: Compiling problem (target solver=ECOS).
(CVXPY) Mar 20 09:52:49 AM: Reduction chain: Dcp2Cone -> CvxAttr2Constr ->
ConeMatrixStuffing -> ECOS
(CVXPY) Mar 20 09:52:49 AM: Applying reduction Dcp2Cone
(CVXPY) Mar 20 09:52:49 AM: Applying reduction CvxAttr2Constr
(CVXPY) Mar 20 09:52:49 AM: Applying reduction ConeMatrixStuffing
(CVXPY) Mar 20 09:52:49 AM: Applying reduction ECOS
(CVXPY) Mar 20 09:52:49 AM: Finished problem compilation (took 1.229e-02
seconds).
-------------------------------------------------------------------------------
                                Numerical solver
-------------------------------------------------------------------------------
(CVXPY) Mar 20 09:52:49 AM: Invoking solver ECOS  to obtain a solution.
-------------------------------------------------------------------------------
                                    Summary
-------------------------------------------------------------------------------
(CVXPY) Mar 20 09:52:49 AM: Problem status: optimal
(CVXPY) Mar 20 09:52:49 AM: Optimal value: 8.524e+02
(CVXPY) Mar 20 09:52:49 AM: Compilation took 1.229e-02 seconds
(CVXPY) Mar 20 09:52:49 AM: Solver (including time spent in interface) took
3.183e-03 seconds
```

```
852.4305773711759

Optimal a:  [ 0.15062914 -0.91314802 -1.52389243 -0.4642144   4.82133807]
Optimal b: [-8.80471718]
```

Predicting

First we load the test data. As above, we make a matrix Y_{test}.

[5]:
```
Xtest = mat['features_test']
ytest = mat['labels_test']
mtest, ntest = Xtest.shape
Ytest = np.zeros((mtest,mtest),float)
for i in range(mtest):
    Ytest[i][i] = ytest[i][0]
```

We only need to find which side of the hyperplane $\{x \mid x^\top a = b\}$ the test data points are - this is obtained by checking whether $y_j = sgn(x_j^\top a - b)$, or equivalently, $y_j \cdot (x_j^\top a - b) > 0$. So again we consider the vector $Y_{\text{test}}(X_{\text{test}}a - b\mathbf{1})$ and find out how many of them have non-positive entries - the lower this number, the better is the prediction.

[6]:
```
print(sum(Ytest@(Xtest@a1.value-b1.value)<=0))
print(sum(Ytest@(Xtest@a2.value-b2.value)<=0))
print(sum(Ytest@(Xtest@a3.value-b3.value)<=0))
```

```
1
2
2
```

[7]:
```
a1.value
```

[7]:
```
array([ 0.14105247,  0.18277618, -0.73224986, -0.10977297,  0.38083898])
```