

AMCS CS 212

Numerical Optimization

Assignment 7

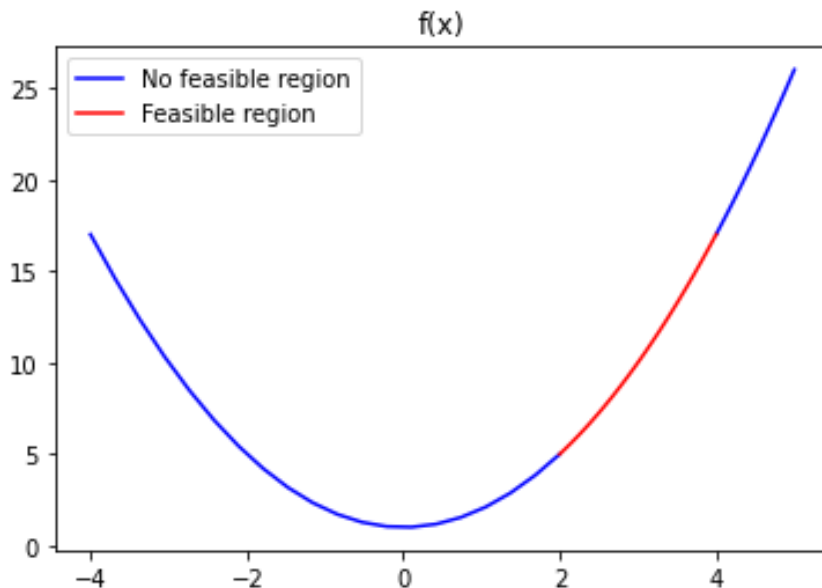
1. Problem in R.

minimize $x^2 + 1$

subject to $(x - 2)(x - 4) \leq 0$

- Derive an expression for the dual function $g(\lambda)$. Plot $g(\lambda)$.

Analyzing the function $f(x)$ with the constrain, we can realized that the feasible region is for $2 \leq X \leq 4$, then the minimum is in $f^*(2)=5$. Thus, we are sure about what to hope from the dual problem.



Applying the first KKT condition and substituting the x value in $L(x, \lambda)$

$$g(\lambda) = \frac{-\lambda^2 + 9\lambda + 1}{\lambda + 1}$$

For solving the problem, we applied the first KKT condition where the gradient respect λ is zero.

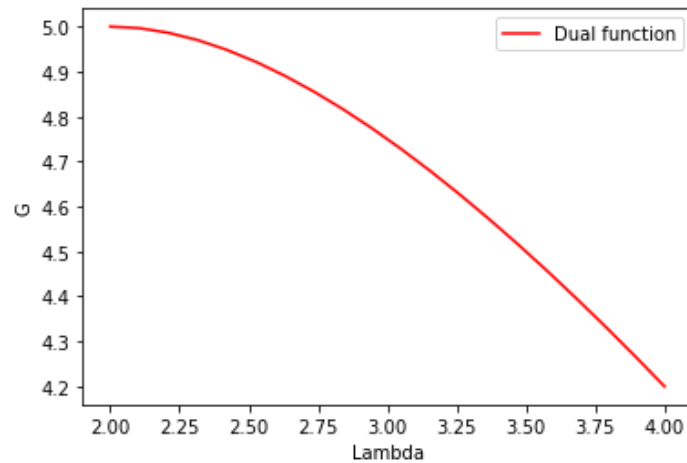
$$\frac{dg(\lambda)}{d\lambda} = \frac{-\lambda^2 - 2\lambda + 8}{(\lambda + 1)^2} = 0, \quad \text{then } \lambda_1 = 2, \lambda_2 = -4.$$

The only feasible point is $\lambda = 2$, then $g^*(2) = 5$

$$g^*(\lambda) = f^*(x)$$

Therefore, the problem has a strong duality

In the next figure we can verify the behavior of $g(\lambda)$ for $\lambda > 2$ (Feasible region). It is evident that the maximum is in $g(\lambda=2)=5$ equal to the $f(x)$ minimum.



2. Dual of a quadratic problem.

(2)

\mathcal{D} is the max of Lagrangian with respect to x

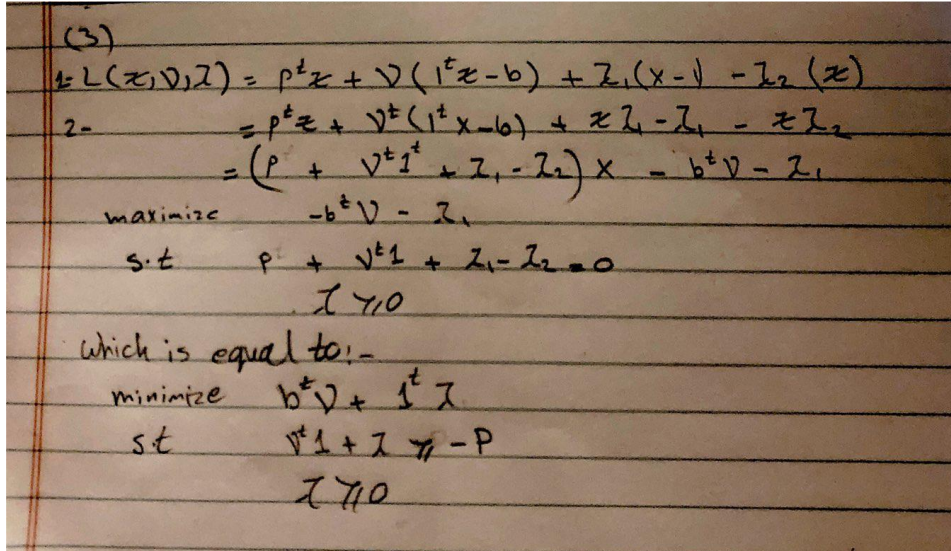
$$\begin{aligned} L(x, v) &= \psi_0(x) + \sum_i v_i h_i(x) \\ &= \psi_0(x) + v^T h(x) \\ &= x^T x + v(Ax - b) \end{aligned}$$

to minimize L over x , $\nabla = 0$

$$\nabla_x L(x, v) = 2x + A^T v = 0 \Rightarrow x = -\frac{1}{2} A^T v$$

$$\begin{aligned} g(v) &= L\left(-\frac{1}{2} A^T v, v\right) = \left(-\frac{1}{2} A^T v\right)^T \left(-\frac{1}{2} A^T v\right) + v\left(A\left(-\frac{1}{2} A^T v\right) - b\right) \\ &= \left(-\frac{1}{2} A^T v\right)^T \left(-\frac{1}{2} A^T v\right) + \\ &= \frac{1}{4} A^T A v^T v - \frac{1}{2} A^T A v^T v - b^T v \\ &= -\frac{1}{4} A^T A v^T v - b^T v \\ &= -\frac{1}{4} v^T A A^T v - b^T v \end{aligned}$$

3. Dual in R^n



(3)
 $L(x, \lambda, \mu) = p^T x + \lambda^T (1^T x - b) + \mu_1 (x - 1) - \mu_2 (x)$
 $= p^T x + \lambda^T (1^T x - b) + x \mu_1 - \mu_1 - x \mu_2$
 $= (p + \lambda^T 1 + \mu_1 - \mu_2)^T x - b^T \lambda - \mu_1$
 maximize $-b^T \lambda - \mu_1$
 s.t. $p + \lambda^T 1 + \mu_1 - \mu_2 = 0$
 $\lambda \geq 0$
 which is equal to:-
 minimize $b^T \lambda + 1^T \mu$
 s.t. $\lambda^T 1 + \mu = p$
 $\lambda \geq 0$

4. Piecewise-linear minimization.

Consider the problem:
 minimize $\max(a_i^T x - b_i)$

- Show that it can be expressed as the smooth linear problem:

$$\begin{aligned} &\text{minimize } t \\ &\text{subject to} \\ &[A \ -1] \begin{bmatrix} x \\ t \end{bmatrix} \leq b \end{aligned}$$

For solving this problem we can define a new variable t and specify that:

$$t \geq \max(a_i^T x - b_i).$$

Now, if we want to minimize t , it will have a lower bound in $t = \max(a_i^T x - b_i)$, so, if t is minimized t will reach its lower bound. On the other hand, the upper bound of $\max(a_i^T x - b_i)$ is t , therefore, if $\max(a_i^T x - b_i)$ is minimized, then $\text{minimize}(\max(a_i^T x - b_i)) = \text{minimize } t$, since they are bound of each other.

Finally, if

$$t \geq \max(a_i^T x - b_i), \text{ then}$$

$$t \geq a_i^T x - b_i \text{ for all } i.$$

Thus, the next inequalities are satisfied.

$$a_i^T x - t \leq b_i$$

Equivalent to:

$$[A \ -1] \begin{bmatrix} x \\ t \end{bmatrix} \leq b$$

With this is shown that the problem can be expressed as:

$$\begin{aligned} & \text{minimize } t \\ & \text{subject to} \\ & [A \quad -\mathbf{1}] \begin{bmatrix} x \\ t \end{bmatrix} \leq b \end{aligned}$$

- Show that the dual may be written as:

$$\begin{aligned} & \text{maximize } -b^t \lambda \\ & \text{subject to } A^t \lambda = 0 \\ & \mathbf{1}^t \lambda = 1 \\ & \lambda \geq 0 \end{aligned}$$

For showing that we generate our Lagrangian as:

$$L(x, t, \vec{\lambda}) = \lambda^T \left[[A \quad -\mathbf{1}] \begin{bmatrix} x \\ t \end{bmatrix} - b \right] + [0 \quad 1] \begin{bmatrix} x \\ t \end{bmatrix}$$

Applying the KKT conditions

$$\nabla_{x,t} L(x, t, \vec{\lambda}) = \lambda^T [A \quad -\mathbf{1}] + [0 \quad 1] = 0$$

Then

$$\lambda^T A = \lambda A^T = 0 \quad \text{and} \quad \mathbf{1}^T \lambda = 1$$

Substituting in the Lagrangian, then

$$g(\lambda) = -\lambda^T b = -\lambda b^T$$

Finally the dual problem is defined as:

$$\begin{aligned} & \text{maximize } -b^t \lambda \\ & \text{subject to } A^t \lambda = 0 \\ & \mathbf{1}^t \lambda = 1 \\ & \lambda \geq 0 \end{aligned}$$

- Write the KKT conditions of the dual problem.

The KKT conditions are given by:

$$\nabla L(\lambda, v_1, v_2) = b + v_1^T [A \quad -\mathbf{1}] + v_2 = 0$$

$$A^t \lambda = 0; \mathbf{1}^t \lambda = 1; \lambda \geq 0$$

$$v_2 I \lambda = 0$$

$$v_2 \geq 0$$

5. Distance between polyhedra.

```
xc = cp.Variable(2)
xd=cp.Variable(2)

I= np.identity(2)
# Construct the problem.

objective = cp.Minimize(cp.quad_form(xc-xd, I))
constraints = [A1@xc <= b1,A2@xd <= b2]
prob = cp.Problem(objective, constraints)
# The optimal objective val, returned by `prob.solve()`
result = prob.solve()
# The optimal value for x is stored in `x.value`.
print('The optimal value xc is ',xc.value)
print('The optimal value xd is ',xd.value)
# The optimal Lagrange multiplier for first constraint
print('The lagrange multipliers are: ')
print(constraints[0].dual_value)

x,y=np.meshgrid(np.linspace(-3,6,150),np.linspace(-3,6,150))

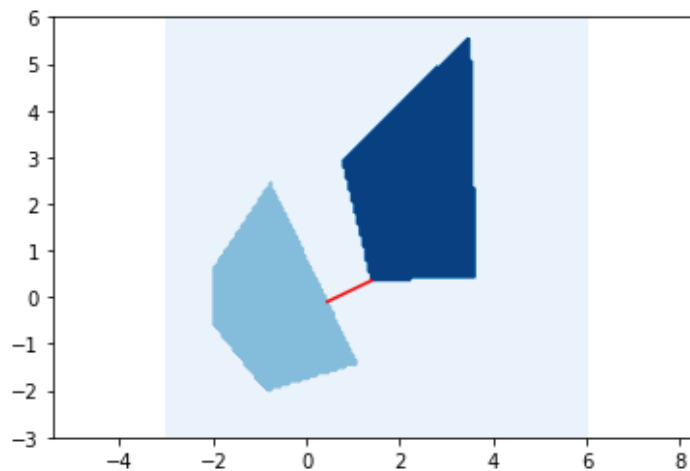
bo1 = np.zeros(np.shape(y))
s=(A1[0,0]*x+A1[0,1]*y<b1[0])\
&(A1[1,0]*x+A1[1,1]*y<b1[1])\
&(A1[3,0]*x+A1[3,1]*y<b1[3])\
&(A1[4,0]*x+A1[4,1]*y<b1[4])\
&(A1[5,0]*x+A1[5,1]*y<b1[5])
bo1[s] = 1

bo2 = np.zeros(np.shape(y))
r=(A2[0,0]*x+A2[0,1]*y<b2[0])\
&(A2[1,0]*x+A2[1,1]*y<b2[1])\
&(A2[3,0]*x+A2[3,1]*y<b2[3])\
&(A2[4,0]*x+A2[4,1]*y<b2[4])
bo2[r] = 2

plt.figure()
plt.contourf(x,y,bo1+bo2,cmap='Blues')
plt.plot([xc.value[0], xd.value[0]],[xc.value[1],xd.value[1]],'r')
plt.axis('equal')
plt.show()
```

Solving the problem we found that the optimal point for minimum distance between the polyhedra are $(x_1, y_1) = (0.4487, -0.0971)$ and $(x_2, y_2) = (1.3963, 0.3567)$. Moreover, we can prove this graphically by the next plot. Printing the multiplier and observing the figure, we can realize that only one constrain of the first polyhedra and two of the second one are active, fact that fits with the results

Multipliers are $\lambda_1 = [0 \ 0.7942 \ 0 \ 0 \ 0 \ 0]$, $\lambda_2 = [0.1985 \ 0 \ 0 \ 0 \ 0.7418]$



6. Classification.

- When the two sets are separable by a hyperplane $ax - b$, the problem can be formulated as that of finding the “thickest slab” that achieves the separation. Formulate this problem as a quadratic optimization problem of the form:

$$\begin{aligned} &\text{minimize } a^T a \\ &\text{subject to } D(Xa - b) \geq 1 \end{aligned}$$

and use `cvxpy` or `scipy.optimize.minimize` to solve it for the data posted. Show your solution as the pair $ax - b = \pm 1$.

```
import cvxpy as cp
from numpy import linalg as la
D = np.identity((m))
for j in range (mblue,m):
    D[j,j] = -1

a0 = cp.Variable(shape=(2,1))
b0 = cp.Variable(1)
oneVec= np.ones((m,1))
#b0 = b0*oneVec
#print(b0)
equation = cp.norm(a,2)

constraints = [(D@(X@a0 - (b0*oneVec))) >=oneVec]
prob = cp.Problem(cp.Minimize(equation),constraints)
prob.solve()

print(a0.value)
print(b0.value)
```

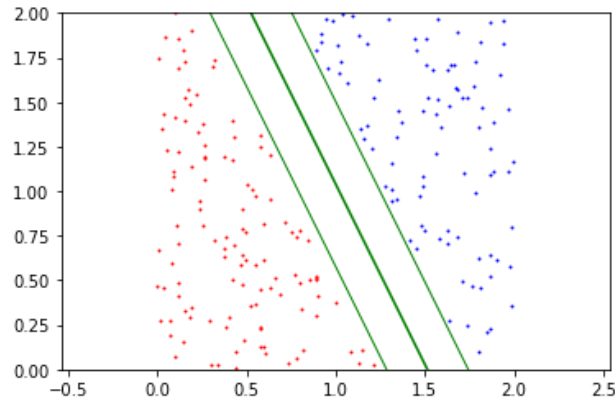


```
# Define x and y values
plt.figure('Test data')
plt.plot(X[0:mblue, 0], X[0:mblue, 1], 'bo', markersize=1)
plt.plot(X[mblue:m, 0], X[mblue:m, 1], 'ro', markersize=1)

print()
a1 = a0.value
b1 = b0.value

plt.plot([0.0, b1/a1[0]], [b1/a1[1], 0.0], color='g')
plt.plot([0.0, (b1+1)/a1[0]], [(b1+1)/a1[1], 0.0], color='g', lw = 1)
plt.plot([0.0, (b1-1)/a1[0]], [(b1-1)/a1[1], 0.0], color='g', lw = 1)

plt.axis('equal')
plt.xlim( 0.0, 2.0 )
plt.ylim( 0.0, 2.0 )
plt.show()
```



- Add a data point that prevents the sets from being separable and verify (with `scipy.optimize` or `cvxpy`) that the problem above becomes infeasible.

```
#the problem becomes infeasible.

import cvxpy as cp
from numpy import linalg as la
D = np.identity((m))
for j in range (mblue,m):
    D[j,j] = -1

a0 = cp.Variable(shape=(2,1))
b0 = cp.Variable(1)
oneVec = np.ones((m,1))
#b0 = b0*oneVec
#print(b0)
equation = cp.norm(a,2)

constraints = [(D@(X@a0 - (b0*oneVec))) >= oneVec]
prob = cp.Problem(cp.Minimize(equation),constraints)
prob.solve()

print(a0.value)
print(b0.value)
```

None
None

- In general the two sets of points cannot be separated by a hyperplane, so we seek to find the best classifier that maximizes the width of a slab that separates the two point sets while minimizing the amount of misclassification (as measured, for example, by the violation of the constraints above). A weighted combination of these objectives can be used. Formulate the problem as:

$$\begin{aligned} &\text{Minimize} \quad 0.5a^t a + \alpha \mathbf{1}^t u \\ &\text{subject to} \quad D(X_a - b\mathbf{1}) \geq \mathbf{1} - u \\ &\quad \quad \quad u \geq 0 \end{aligned}$$

and solve it to generate the tradeoff curve of Pareto optimal points for the given data. Comment on the shape of the curve. Plot the solution. Does it make sense?

```
#3
import cvxpy as cp
from numpy import linalg as la
D = np.identity((m))
for j in range (mblue,m):
    D[j,j] = -1

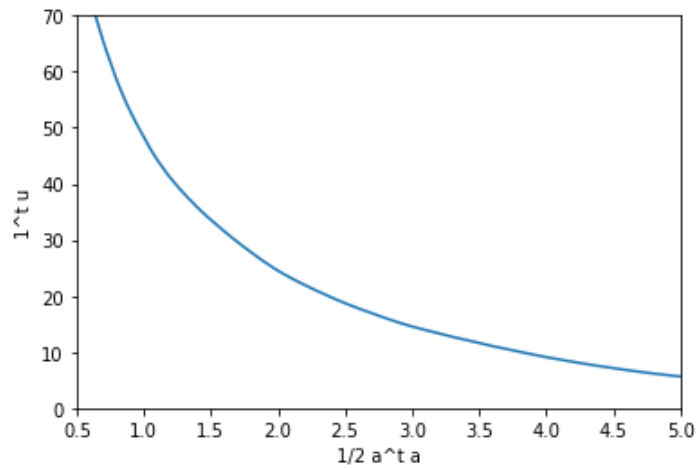
a0 = cp.Variable(shape=(2,1))
b0 = cp.Variable(1)
u = cp.Variable((m,1))
oneVec = np.ones((m,1))
#b0 = b0*oneVec
#print(b0)
alpha = 1e-5

sols=[]
sum_u=[]
for alpha in np.arange(0,2,0.001):
    equation = 0.5*(cp.norm(a0,2))**2 + (np.transpose(alpha*oneVec)@ u)
    constraints = [(D@(X@a0 - (b0*oneVec))) >= oneVec - u , u>=0]
    prob = cp.Problem(cp.Minimize(equation),constraints)
    sol = prob.solve()
    a1 = ((1/2)*(np.transpose(a0.value)@a0.value))
    sols.append(a1[0,0])
    u1 = ((np.transpose(oneVec)@u.value))
    sum_u.append(u1[0,0])

print(u.value.shape)

plt.plot(sols,sum_u)

#plt.axis('equal')
plt.xlim( 0.5, 5 )
plt.ylim( 0.0, 70.0 )
plt.xlabel("1/2 a^t a")
plt.ylabel("1^t u")
plt.show
print('the curve make sense since we dont have best value for alpha.')
```

The curve make sense because the optimization problem depends on two different functions. Then, when one is optimized, the other could have large value and viceversa

```
# Define x and y values
plt.figure('Test data')
plt.plot(X[0:mblue, 0], X[0:mblue, 1], 'bo', markersize=1)
plt.plot(X[mblue:m, 0], X[mblue:m, 1], 'ro', markersize=1)

print()
a2 = a0.value
b2 = b0.value

plt.plot([0.0, b2/a2[0]], [b2/a2[1], 0.0], color='g')
plt.plot([0.0, (b2+1)/a2[0]], [(b2+1)/a2[1], 0.0], color='g', lw = 1)
plt.plot([0.0, (b2-1)/a2[0]], [(b2-1)/a2[1], 0.0], color='g', lw = 1)

#plt.plot([0.0, b/a[0]], [b/a[1], 0.0], color='y')
#plt.plot([0.0, (b+1)/a[0]], [(b+1)/a[1], 0.0], color='y', lw = 1)
#plt.plot([0.0, (b-1)/a[0]], [(b-1)/a[1], 0.0], color='y', lw = 1)

plt.axis('equal')
plt.xlim( 0.0, 2.0 )
plt.ylim( 0.0, 2.0 )
plt.show()
```

