Curso de Optimización (DEMAT)

Tarea 11

Leslie Janeth Quincosa Ramírez

Descripción:	Fechas
Fecha de publicación del documento:	**Mayo 13, 2022**
Fecha límite de entrega de la tarea:	**Mayo 22, 2022**

Indicaciones

- Envie el notebook que contenga los códigos y las pruebas realizadas de cada ejercicio.
- Si se requiren algunos scripts adicionales para poder reproducir las pruebas, agreguelos en un ZIP junto con el notebook.
- Genere un PDF del notebook y envielo por separado.

Ejercicio 1 (3 puntos)

Usando alguna librería de Python para resolver problemas de programación lineal, escriba y resuelva el problema de la Tarea 10:

$$
\begin{aligned}
\max \quad & x_1 + x_2 \\
& 50x_1 + 24x_2 \leq 2400 \\
& 30x_1 + 33x_2 \leq 2100 \\
& x_1 \geq 45 \\
& x_2 \geq 5
\end{aligned}
$$

1. Cambie el problema para que todas las desigualdes sean de la forma

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}.$$

1. Construya los vectores $\mathbf{b}, \mathbf{c}$
   y la matriz $\mathbf{A}$
   y resuelva el problema con la librería.
2. Imprima un mensaje que indique si se encontró la solución, y en ese caso imprima :

- la solución $\mathbf{x}$
  ,
- el valor de la función objetivo,
- las variables de holgura,

1. Calcule los errores

$$E_x = \sum_{x_i<0} |x_i| .$$

$$E_{b-Ax} = \sum_{(b-Ax)_i<0} |(b-Ax)_i|$$

Es decir, se suman las componentes de $\mathbf{x}$
que no cumplen la condición $\mathbf{x} \geq 0$
y las componentes que no cumplen con $\mathbf{A}\mathbf{x} \leq \mathbf{b}$

.

1. **Defina la tolerancia** $\tau = \sqrt{\epsilon_m}$

   , donde ϵ_m

   es el épsilon de la máquina. Si $E_x < \tau$

   imprima un mensaje de que se cumple la condición de no negatividad, y si $E_{b-Ax} < \tau$

   imprima un mensaje de que se cumplen las restricciones de desigualdad.

Solución:

In [75]:

```
pip install pulp
```

```
Requirement already satisfied: pulp in /usr/local/lib/python3.7/dist-packages (2.6.0)
```

In [76]:

```
from scipy.optimize import linprog
import scipy
import numpy as np
import sys
```

In [77]:

```
#Coficientes de la función a minimizar
c = np.array([-1, -1, 0, 0, 0, 0 ])

#Restricciones
b = np.array([2400, 2100, 45, 5])
A = np.array([[50, 24, 1, 0, 0, 0],
              [30, 33, 0, 1, 0, 0],
              [ 1,  0, 0, 0,-1, 0],
              [ 0,  1, 0, 0, 0,-1] ])

from pulp import LpMaximize, LpProblem, LpStatus, lpSum, LpVariable

# Creacción de una instancia de la  clase
model = LpProblem(name="small-problem", sense=LpMaximize)

# Variables de decisión
x1 = LpVariable(name="x1", lowBound=45)
x2 = LpVariable(name="x2", lowBound=5)

# Se agregan las restricciones del modelo
model += (50*x1 + 24*x2 <= 2400, "R1")
model += (30*x1 + 33*x2  <= 2100, "R2")


# Función objetivo
model += x1 + x2

print(model)
```

```
small-problem:
MAXIMIZE
1*x1 + 1*x2 + 0
SUBJECT TO
R1: 50 x1 + 24 x2 <= 2400

R2: 30 x1 + 33 x2 <= 2100

VARIABLES
45 <= x1 Continuous
5 <= x2 Continuous
```

In [78]:

```python
# Cálculo de la solución
status = model.solve()

print("Resultado: ", model.status, " | ", LpStatus[model.status])

print("Valor de la funciónn objetivo: " , model.objective.value())


print('Solución:')
for var in model.variables():
    print("%10s: %f"  % (var.name, var.value()) )

print('\nVariables de holgura:')
for name, constraint in model.constraints.items():
    print("%10s: %f" % (name, constraint.value()) )
```

```
Resultado:  1  |  Optimal
Valor de la funciónn objetivo:  51.25
Solución:
        x1: 45.000000
        x2: 6.250000

Variables de holgura:
        R1: 0.000000
        R2: -543.750000
```

In [79]:

```python
# La función que vamos a utilizar resuelve un problema de minimización,
# por lo que hay que cambiar de signo a los coeficientes de la función objetivo:

# Coeficientes de la funcion objetivo
obj = [-1, -1]

# Coeficientes del lado izquierdo de las desigualdades del tipo "menor o igual a"
lhs_ineq = [[50, 24],
            [30, 33]]

A = np.array([[50, 24],[30, 33]])

# Coeficientes del vector del lado derecho de las desigualdades del tipo "menor o igual a
"
rhs_ineq = [2400, 2100]
b = np.array([2400, 2100])

# Cotas de las variables
bnd = [(45, scipy.inf),  # cotas para x1
       (5, scipy.inf)]  # cotas para x2

opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq, bounds=bnd, method="simplex")

print('\nResultado del proceso:', opt.message)
if opt.success:
    print('Valor de la función objetivo:', opt.fun)
    print('Solución:\n', opt.x)
    print('\nVariables de holgura:\n', opt.slack)
```

```
Resultado del proceso: Optimization terminated successfully.
Valor de la función objetivo: -51.25
Solución:
 [45.    6.25]

Variables de holgura:
 [  0.   543.75]
```

In [80]:

```python
#Calcular el error
eps = sys.float_info.epsilon
tol = eps**(1/2)
```

```
Error_x = 0
for i in range(len(opt.x)):
  if opt.x[i] < 0:
    Error_x = Error_x + np.absolute(opt.x[i])
print('Error_x:', Error_x)
if Error_x < tol:
  print('Se cumple la condición de no negatividad')

Error_bAx = 0
for i in range(len(opt.x)):
  if (b - A@opt.x)[i] < 0:
    Error_bAx = Error_bAx + np.absolute((b - A@opt.x)[i])
print('Error_b-Ax:', Error_bAx)

if Error_bAx < tol:
  print('se cumplen las restricciones de desigualdad')
```

```
Error_x: 0
Se cumple la condición de no negatividad
Error_b-Ax: 0
se cumplen las restricciones de desigualdad
```

En este ejercicio implemente ambas librerías: pulp y scipy

Ejercicio 2 (3 puntos)

1. **Escriba el problema anterior en su forma estándar.**
2. **Construya los vectores** b, c **y la matriz** A **y resuelva este problema con la librería.**
3. **Imprima un mensaje que indique si se encontró la solución, y en ese caso imprima la solución, el valor de la función objetivo, las variables de holgura y el error**

$$\|\mathbf{Ax} - \mathbf{b}\|.$$

1. **Calcule el error** E_x **como en el Ejercicio 1 y si** $E_x < \tau$ **imprima un mensaje de que se cumple la condición de no negatividad.**

Solución:

Forma estándar:

$$\begin{aligned}
\min \quad & -x_1 - x_2 \\
& 50x_1 + 24x_2 + x_3 = 2400 \\
& 30x_1 + 33x_2 + x_4 = 2100 \\
& x_1 - x_5 = 45 \\
& x_2 - x_6 = 5 \\
& x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{aligned}$$

In [80]:

```

```

In [81]:

```
#Coficientes de la función a minimizar
c = np.array([-1, -1, 0, 0, 0, 0 ])

#Restricciones
```

```
b = np.array([2400, 2100, 45, 5])

A = np.array([[50, 24, 1, 0, 0, 0],
              [30, 33, 0, 1, 0, 0],
              [ 1,  0, 0, 0,-1, 0],
              [ 0,  1, 0, 0, 0,-1] ])

# Cotas de las variables
bnd = (0, None)

opt = linprog(c = c, A_eq = A, b_eq = b, bounds=bnd, method="simplex")

print('\nResultado del proceso:', opt.message)
if opt.success:
    print('¡Se encontró la solución!')
    print('Valor de la función objetivo:', opt.fun)
    print('Solución:\n', opt.x)
    print('\nVariables de holgura:\n', opt.slack)
    print('|| Ax - b||: ', np.linalg.norm(A@opt.x - b))

xval = opt.x
print(xval)
Error_x = 0
for i in range(len(opt.x)):
  if opt.x[i] < 0:
    Error_x = Error_x + np.absolute(opt.x[i])
print('Error_x:', Error_x)
if Error_x < tol:
  print('Se cumple la condición de no negatividad')
```

```
Resultado del proceso: Optimization terminated successfully.
¡Se encontró la solución!
Valor de la función objetivo: -51.25
Solución:
 [ 45.      6.25    0.    543.75    0.      1.25]

Variables de holgura:
 []
|| Ax - b||:  3.552713678800501e-15
[ 45.      6.25    0.    543.75    0.      1.25]
Error_x: 0
Se cumple la condición de no negatividad
```

Ejercicio 3 (4 puntos)

1. **Escriba el problema dual del Ejercicio 2.**
2. **Resuelva el problema dual con la librería. Esto debería devolver el vector λ que son los multiplicadores de Lagrange de la restricciones de igualdad del problema primal.**
3. **Imprima un mensaje que indique si se encontró la solución, y de ser así, imprima λ, el valor de la función objetivo y las variables de holgura.**
4. **Usando el valor $\mathbf{x}$ del Ejercicio 2, imprima el error relativo**

$$\frac{|\mathbf{c}^\top \mathbf{x} - \mathbf{b}^\top \lambda|}{|\mathbf{c}^\top \mathbf{x}|}.$$

1. **Defina el vector $\mathbf{s}$ como las variables de holgura.**
2. **Programe una función que reciba los vectores $\mathbf{b}, \mathbf{c}$, $\mathbf{x}, \lambda, \mathbf{s}$, la matriz A y una tolerancia τ**

, y verique que se cumplen las condiciones KKT:

$$\begin{aligned} \mathbf{A}^\top \lambda + \mathbf{s} &= \mathbf{c}, && (1)\\ \mathbf{A}\mathbf{x} &= \mathbf{b}, && (2)\\ \mathbf{x} &\geq \mathbf{0}, && (3)\\ \mathbf{s} &\geq \mathbf{0}, && (4)\\ x_i s_i &= 0, \quad i = 1, 2, \ldots, n. && (5) \end{aligned}$$

Calcule los errores E_x
y E_s
como en el Ejercicio 1, para saber que tanto se violan las restricciones $\mathbf{x} \geq \mathbf{0}$
y $\mathbf{s} \geq \mathbf{0}$
.

La función debe imprimir

- **El error $\|\mathbf{A}^\top \lambda + \mathbf{s} - \mathbf{c}\|$**
  .
- **El error $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|$**
  .
- **Si $E_x < \tau$**
  , imprima que se cumple las restricciones de no negatividad de $\mathbf{x}$
  .
- **Si $E_s < \tau$**
  , imprima que se cumple las restricciones de no negatividad de $\mathbf{s}$
  .
- **Calcule el valor de la suma $\sum_i |x_i s_i|$**
  y si es menor que τ
  , imprima un mensaje que indique que se cumple la condición de complementariedad.

1. **Use la función anterior en el problema para reportar los resultados.**

> **Nota: En el problema dual las variables en λ**
> **no tienen restricciones de cota. Si usa, por ejemplo, la función `linprog` para resolver el problema, ponga explícitamente que las cotas de las variables son $-\infty$**
> **e ∞**
> **para que la función no use las cotas que tiene fijas de manera predeterminada.**

Primero escribamos el problema primal, del ejercicio 2.

$$\begin{aligned} \max \quad & x_1 + x_2 \\ & 50x_1 + 24x_2 \leq 2400 \\ & 30x_1 + 33x_2 \leq 2100 \\ & x_1 \geq 45 \\ & x_2 \geq 5 \end{aligned}$$

Enotonces, el problema dual es

$$\begin{aligned} \max \quad & 2400\lambda_1 + 2100\lambda_2 + 45\lambda_3 + 5\lambda_4 \\ & 50\lambda_1 + 30\lambda_2 - \lambda_3 \leq 1 \\ & 24\lambda_1 + 33\lambda_2 - \lambda_4 \leq 1 \\ & \lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_3 \geq 0, \lambda_4 \geq 0 \end{aligned}$$

Problema estándar del dual

$$\begin{aligned} \min \quad & -2400\lambda_1 - 2100\lambda_2 - 45\lambda_3 - 5\lambda_4 \\ & 50\lambda_1 + 30\lambda_2 + \lambda_3 = 1 \end{aligned}$$

$$24\lambda_1 + 33\lambda_2 + \lambda_4 = 1$$

$$\lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_3 \leq 0, \lambda_4 \leq 0$$

In [82]:

```
#Coficientes de la función a minimizar
b = np.array([2400, 2100, 45, 5])

#Restricciones
c = np.array([-1, -1, 0, 0, 0, 0 ])

A = np.array([[50, 24, 1, 0, 0, 0],
              [30, 33, 0, 1, 0, 0],
              [ 1,  0, 0, 0,-1, 0],
              [ 0,  1, 0, 0, 0,-1]])

# Cotas de las variables
bnd = (None, None)


opt = linprog(c = -b, A_ub = A.T, b_ub= c, bounds=bnd)

print('\nResultado del proceso:', opt.message)
if opt.success:
    print('¡Se encontró la solución!')
    print('Valor de la función objetivo:', opt.fun)
    print('Solución del vector lambda:\n', opt.x)
    print('\nVariables de holgura:\n', opt.slack)

lam = opt.x
s = opt.slack
```

```
Resultado del proceso: Optimization terminated successfully.
¡Se encontró la solución!
Valor de la función objetivo: 51.25000000631624
Solución del vector lambda:
 [-4.16666667e-02 -3.26266209e-13  1.08333333e+00  4.25959588e-11]

Variables de holgura:
 [1.21579635e-10 9.83173543e-11 4.16666667e-02 3.26266209e-13
 1.08333333e+00 4.25959588e-11]
```

```
/usr/local/lib/python3.7/dist-packages/numpy/core/fromnumeric.py:86: VisibleDeprecationWa
rning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of list
s-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to
do this, you must specify 'dtype=object' when creating the ndarray.
  return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
```

In [83]:

```
print('Error relativo: ', np.absolute(c.T@ xval - b.T@lam)/np.absolute(c.T@xval))
```

```
Error relativo:  1.2324374149945314e-10
```

In [84]:

```
def condicionesKKT(b, c, x, lamb, s, A, tol):
  for i in range(len(x)):
    if x[i]*s[i] < tol:
      print("Se cumple x[", i, "] * s[", i, "] = 0")
    else:
      print("No se cumple x[", i, "] * s[", i, "] = 0")
  Error_x = 0
  for i in range(len(x)):
    if x[i] < 0:
      Error_x = Error_x + np.absolute(x[i])
  print('Error_x:', Error_x)
  if Error_x < tol:
    print('Se cumple la condición de no negatividad de x')
  Error_s = 0
```

```
  for i in range(len(s)):
    if s[i] < 0:
      Error_s = Error_s + np.absolute(s[i])
    print('Se cumple la condición de no negatividad de s')
  print('Error_s:', Error_s)
  print("Error ||AT*lamb + s -c||:",  np.linalg.norm(A.T@lamb + s - c) )
  print("Error ||Ax - b||:",  np.linalg.norm(A@x - b))
  Suma_xs = 0
  for i in range(len(s)):
    Suma_xs = Suma_xs + np.absolute(x[i]*s[i])
  if Suma_xs < tol:
    print('Se cumple la condición de complementariedad')
```

In [85]:

```
eps = sys.float_info.epsilon
tol = eps**(1/2)

condicionesKKT(b, c, xval, lam, s, A, tol)
```

```
Se cumple x[ 0 ] * s[ 0 ] = 0
Se cumple x[ 1 ] * s[ 1 ] = 0
Se cumple x[ 2 ] * s[ 2 ] = 0
Se cumple x[ 3 ] * s[ 3 ] = 0
Se cumple x[ 4 ] * s[ 4 ] = 0
Se cumple x[ 5 ] * s[ 5 ] = 0
Error_x: 0
Se cumple la condición de no negatividad de x
Se cumple la condición de no negatividad de s
Se cumple la condición de no negatividad de s
Se cumple la condición de no negatividad de s
Se cumple la condición de no negatividad de s
Se cumple la condición de no negatividad de s
Se cumple la condición de no negatividad de s
Error_s: 0
Error ||AT*lamb + s -c||: 0.0
Error ||Ax - b||: 3.552713678800501e-15
Se cumple la condición de complementariedad
```

In [86]:

```
print(b.T@lam)
print(c.T@xval)
```

```
-51.25000000631624
-51.25
```

Notemos que el problema dual y el primal son equivalentes ya que $b^T\lambda = c^T x$ **. Luego, los errores son muy peuqueños y se cumplen las condiciones necesarias para que el problema haya alcanzado el valor óptimo.**