# Homework 9
# Linear Programming

Zhicong Sun
Supervisor: Shuang-Hua Yang
From: Harbin Institute of Technology
2020.11.19

# Exercise 8-1

- **Find an available LP software**

***SciPy*: Scientific computing tools for Python**

SciPy.org

*SciPy* (pronounced "Sigh Pie")
is a Python-based ecosystem of open-source software for mathematics, science, and engineering.
In particular, these are some of the core packages:

**NumPy**
Base N-dimensional array package

**SciPy library**
Fundamental library for scientific computing

**Matplotlib**
Comprehensive 2-D plotting

**IPython**
Enhanced interactive console

**SymPy**
Symbolic mathematics

**pandas**
Data structures & analysis

# Exercise 8-1

■ Find an available LP software

***SciPy*: Scientific computing tools for Python**

SciPy.org | Docs | SciPy v1.5.4 Reference Guide

## Optimization and root finding (scipy.optimize)

SciPy `optimize` provides functions for minimizing (or maximizing) objective functions, possibly subject to constraints. It includes solvers for nonlinear problems (with support for both local and global optimization algorithms), linear programing, constrained and nonlinear least-squares, root finding, and curve fitting.

## scipy.optimize.linprog

scipy.optimize.linprog(*c, A_ub=None, b_ub=None, A_eq=None, b_eq=None, bounds=None, method='interior-point', callback=None, options=None, x0=None*) [source]

Linear programming: minimize a linear objective function subject to linear equality and inequality constraints.

Linear programming solves problems of the following form:

$$\boxed{\begin{aligned} \min_x \ & c^T x \\ \text{such that } & A_{ub} x \leq b_{ub}, \\ & A_{eq} x = b_{eq}, \\ & l \leq x \leq u, \end{aligned}}$$

where $x$ is a vector of decision variables; $c$, $b_{ub}$, $b_{eq}$, $l$, and $u$ are vectors; and $A_{ub}$ and $A_{eq}$ are matrices.

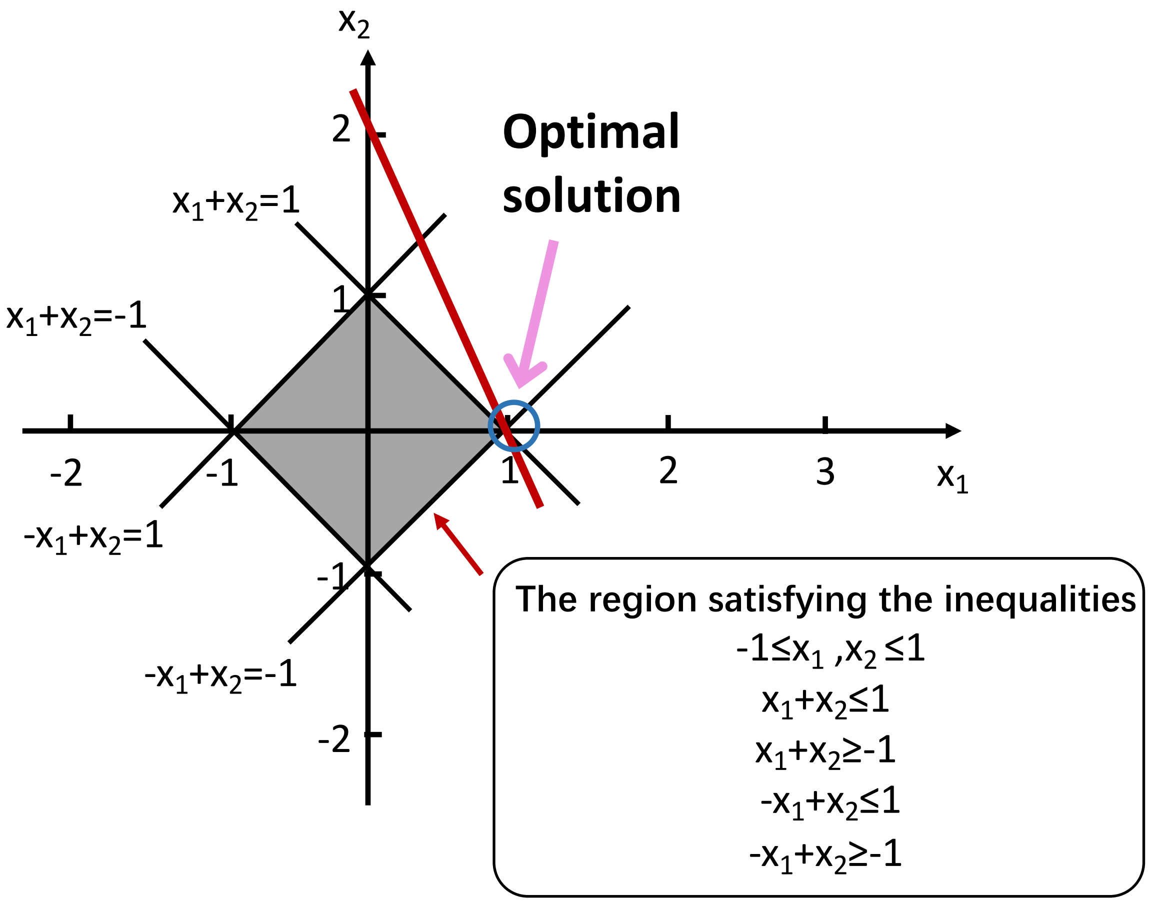
# *Exercise 8-1*

■ **Generate a simple example of the LP problem.**

## *Simple example*

Maximize: $2x_1+x_2$

Subject to: $x_1+x_2\leq 1$, $x_1+x_2\geq -1$, $-x_1+x_2\leq 1$, $-x_1+x_2\geq -1$,



# Exercise 8-1

■ **Generate a simple example of the LP problem.**

***Simple example***

Form: $\min_x c^T x$ **s. t** $\begin{cases} A_{ub}x \le b_{ub} \\ A_{eq}x = b_{eq} \\ l \le x \le u \end{cases}$

Maximize: $2x_1+x_2$ $\Rightarrow$ $\underbrace{\begin{bmatrix} 2 \\ 1 \end{bmatrix}}_{c^T} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

Subject to: $x_1+x_2\le1$, $x_1+x_2\ge-1$, $-x_1+x_2\le1$, $-x_1+x_2\ge-1$

$$\begin{aligned} x_1 + x_2 &\le 1 \\ x_1 + x_2 &\ge -1 \\ -x_1 + x_2 &\le 1 \\ -x_1 + x_2 &\ge -1 \end{aligned} \Rightarrow \begin{aligned} x_1 + x_2 &\le 1 \\ -x_1 - x_2 &\le 1 \\ -x_1 + x_2 &\le 1 \\ x_1 - x_2 &\le 1 \end{aligned} \Rightarrow \underbrace{\begin{bmatrix} 1 & 1 \\ -1 & -1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix}}_{A_{ub}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}}_{x} \le \underbrace{\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}}_{b_{ub}}$$

# Exercise 8-1

- Generate a simple example of the LP problem.

## *Simple example*

$$\underbrace{\begin{bmatrix} 2 \\ 1 \end{bmatrix}}_{c^T} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \underbrace{\begin{bmatrix} 1 & 1 \\ -1 & -1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix}}_{A_{ub}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}}_{x} \leq \underbrace{\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}}_{b_{ub}}$$

```python
import numpy
from scipy import optimize

c = numpy.array([2,1])

A_ub = numpy.array([[1,1],[-1,-1],[-1,1],[1,-1]])
b_ub = numpy.array([1,1,1,1])

res = optimize.linprog(-c,A_ub,b_ub)
print(res)
print("Optimal result is",-res.fun)
```

```
In [25]: runfile('/Users/apple/Desktop/GitHub/Advanced-algorithm/w9_LP/w9_LP_simple.py', wdir='/Users/apple/Desktop/GitHub/Advanced-algorithm/w9_LP')
     con: array([], dtype=float64)
     fun: -2.000000021045033
 message: 'Optimization terminated successfully.'
     nit: 4
   slack: array([-1.42245193e-08,  2.00000001e+00,  2.00000000e+00,
5.83492366e-10])
  status: 0
 success: True
       x: array([1.00000001e+00, 7.40400588e-09])
Optimal result is 2.000000021045033
```

**Position:(1,0) Optimal Result:2**

# Exercise 8-1

- **Solve the large example using the LP software**

## *Complex example*

1,000,000 variables & 10 constraints

```python
from scipy import optimize
import numpy as np
c = np.random.rand(1000000)
A_ub = np.random.rand(10,1000000)
b_ub = np.random.rand(10)

res = optimize.linprog(c,A_ub,b_ub)
print(res)
print("Optimal result is",res.fun)
```

```
In [27]: runfile('/Users/apple/Desktop/GitHub/Advanced-algorithm/w9_LP/
w9_LP_complex.py', wdir='/Users/apple/Desktop/GitHub/Advanced-algorithm/
w9_LP')
     con: array([], dtype=float64)
     fun: 7.981373152761669e-10
 message: 'Optimization terminated successfully.'
     nit: 17
   slack: array([0.09687831, 0.95305721, 0.31397605, 0.0455835 , 0.48556138,
       0.76671492, 0.28367432, 0.90509625, 0.72168116, 0.7162813 ])
  status: 0
 success: True
       x: array([6.73002971e-16, 3.51827499e-15, 3.19594763e-15, ...,
       1.18297812e-15, 8.27759647e-16, 1.63249049e-15])
Optimal result is 7.981373152761669e-10
```

SUSTech Southern University of Science and Technology

# Thanks!

## Please contact me with email if you have any problem

ZhicongSun
Computer Science and Engineering
12032471@mail.sustech.edu.cn
Github: RadarSun
2020.11.19