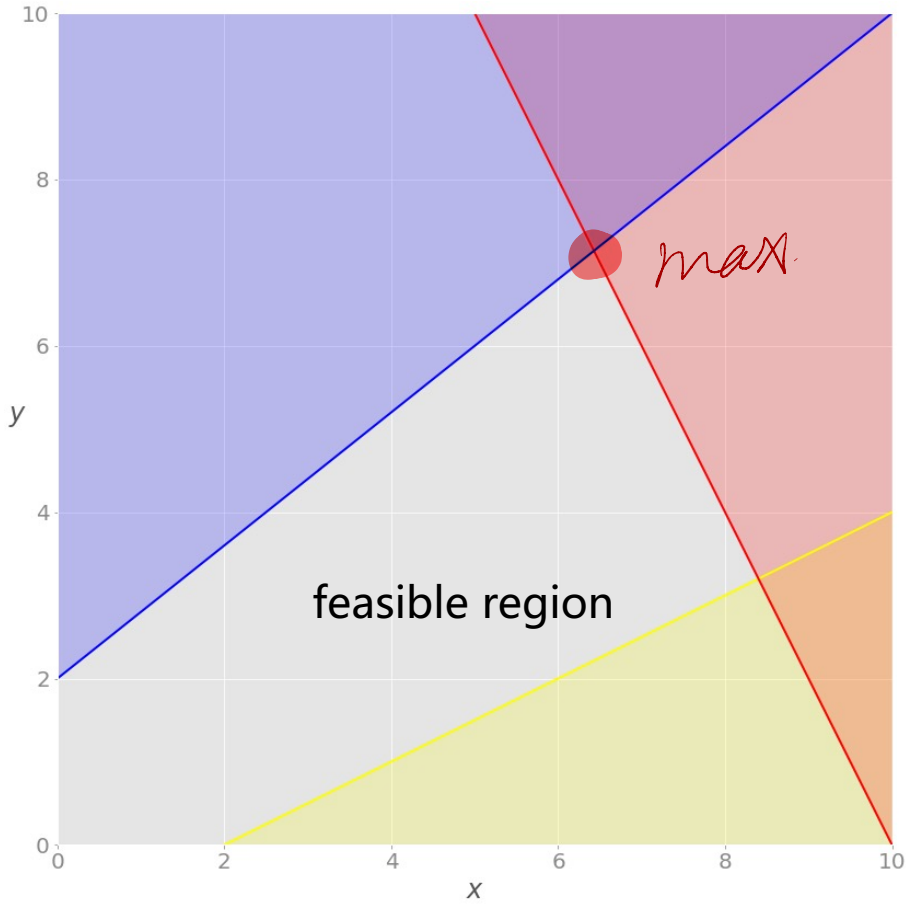
Linear programming examples

2021.04.18

Linear programming

- Linear programming is a technique for the optimization of a **linear** objective function, subject to **linear equality** and **linear inequality** constraints

$$
\begin{aligned}
\text{maximize} \quad & z = x + 2y \\
\text{subject to:} \quad & 2x + y \leq 20 \\
& -4x + 5y \leq 10 \\
& -x + 2y \geq -2 \\
& x \geq 0 \\
& y \geq 0
\end{aligned}
$$

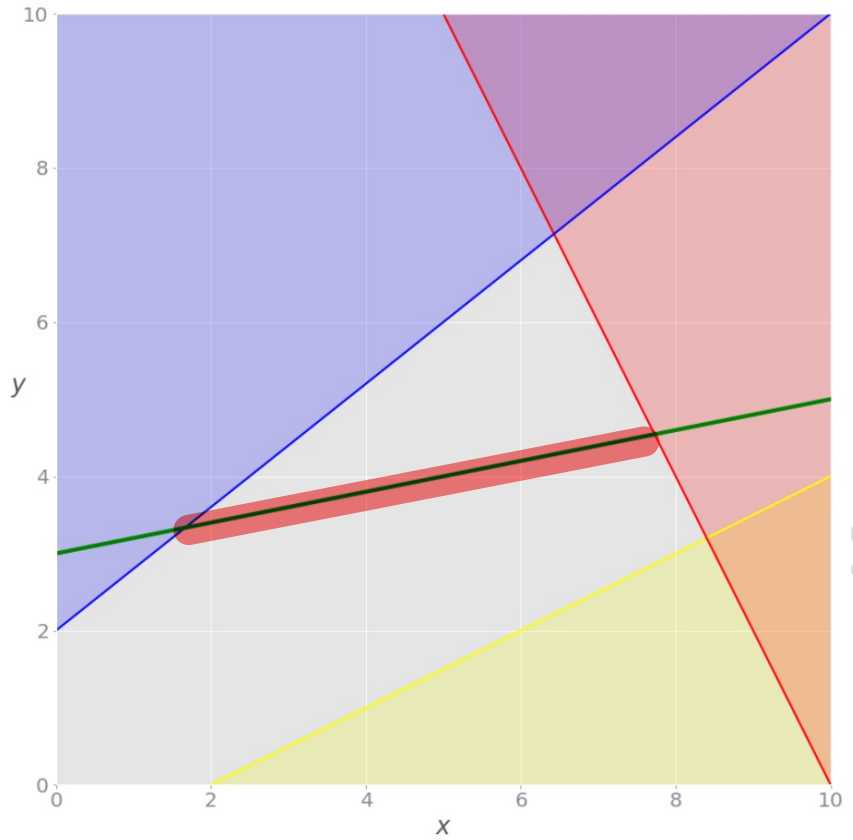


https://en.wikipedia.org/wiki/Linear_programming
https://realpython.com/linear-programming-python/

Linear programming

- Linear programming is a technique for the optimization of a **linear** objective function, subject to **linear equality** and **linear inequality** constraints

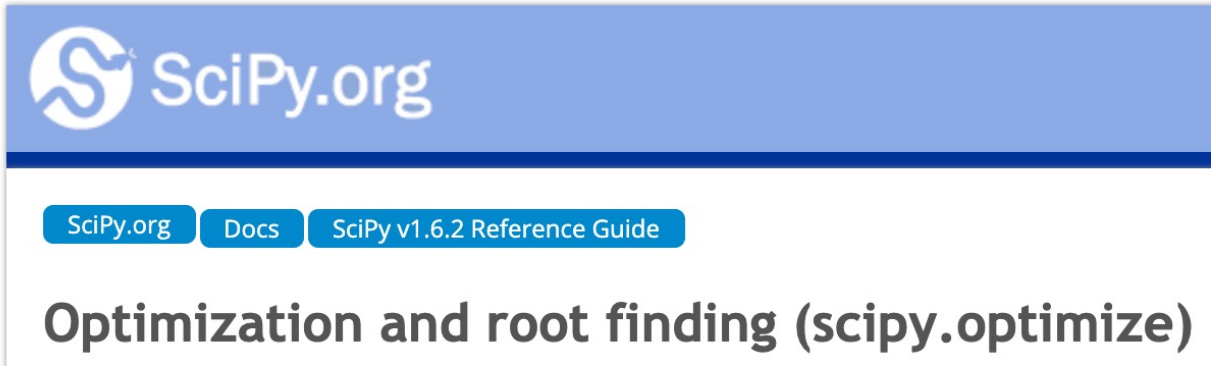
$$
\begin{aligned}
\text{maximize} \quad & z = x + 2y \\
\text{subject to:} \quad & 2x + y \leq 20 \\
& -4x + 5y \leq 10 \\
& -x + 2y \geq -2 \\
& -x + 5y = 15 \\
& x \geq 0 \\
& y \geq 0
\end{aligned}
$$



https://en.wikipedia.org/wiki/Linear_programming
https://realpython.com/linear-programming-python/

Code - scipy

- SciPy is open-source software for mathematics, science, and engineering



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:

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

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.

https://docs.scipy.org/doc/scipy/reference/index.html

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:

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

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.

Alternatively, that's:

minimize:

```
c @ x
```

such that:

```
A_ub @ x <= b_ub
A_eq @ x == b_eq
lb <= x <= ub
```

Note that by default `lb = 0` and `ub = None` unless specified with `bounds`.

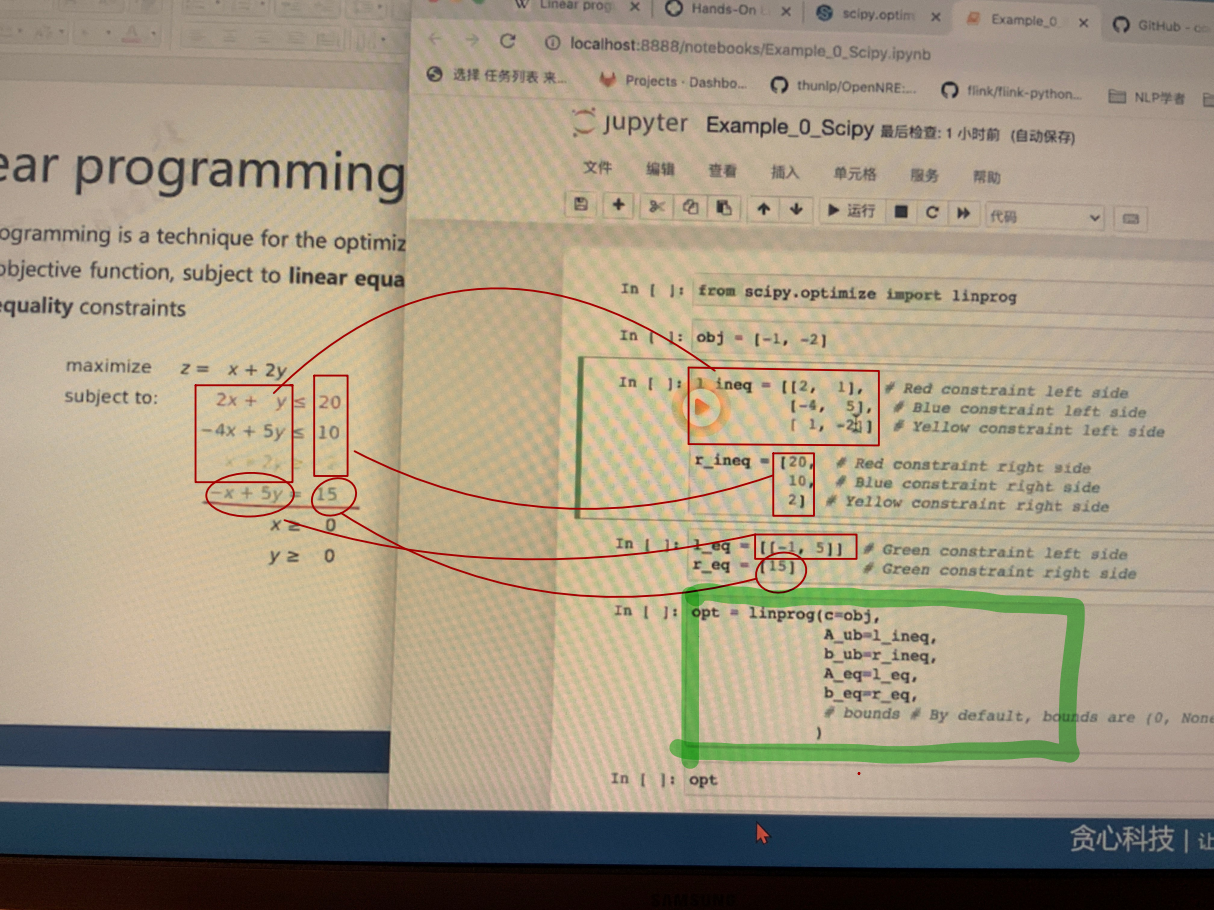
Parameters: `c` : *1-D array*

The coefficients of the linear objective function to be minimized.

`A_ub` : *2-D array, optional*

The inequality constraint matrix. Each row of `A_ub` specifies the coefficients of a linear inequality constraint on `x`.

`b_ub` : *1-D array, optional*


ear programming
ogramming is a technique for the optimiz
bjective function, subject to linear equa
quality constraints
maximize z = x + 2y
subject to:
2x + y ≤ 20
−4x + 5y ≤ 10
−x + 5y = 15
x ≥ 0
y ≥ 0
Linear prog
Hands-On
scipy.optim
Example_0
GitHub - cir
localhost:8888/notebooks/Example_0_Scipy.ipynb
选择 任务列表 来...
Projects · Dashbo...
thunlp/OpenNRE:...
flink/flink-python...
NLP学者
jupyter Example_0_Scipy 最后检查: 1 小时前 (自动保存)
文件 编辑 查看 插入 单元格 服务 帮助
运行
代码
In [ ]: from scipy.optimize import linprog
In [ ]: obj = [-1, -2]
In [ ]: l_ineq = [[2, 1], # Red constraint left side
[-4, 5], # Blue constraint left side
[1, -2]] # Yellow constraint left side
r_ineq = [20, # Red constraint right side
10, # Blue constraint right side
2] # Yellow constraint right side
In [ ]: l_eq = [[-1, 5]] # Green constraint left side
r_eq = [15] # Green constraint right side
In [ ]: opt = linprog(c=obj,
A_ub=l_ineq,
b_ub=r_ineq,
A_eq=l_eq,
b_eq=r_eq,
# bounds # By default, bounds are (0, None
)
In [ ]: opt
贪心科技 | 让

Example 1

- A factory produces 4 different products. The profit per unit of product is $20, $12, $40, and $25 for the first, second, third, and fourth product, respectively
- For each unit of the first product, 3 units of the raw material A are consumed. Each unit of the second product requires 2 units of the raw material A and 1 unit of the raw material B. Each unit of the third product needs 1 unit of A and 2 units of B. Finally, each unit of the fourth product requires 3 units of B
- Due to the transportation and storage constraints, the factory can consume up to 100 units of the raw material A and 90 units of B per day
- The goal is to determine the profit-maximizing daily production amount for each product
- Due to manpower constraints, the total number of units produced per day can't exceed 50

Let the daily produced amount of the first product is x_1, the second product is x_2, and so on

$$3x_1 + 2x_2 + x_3 \leq 100 \quad \text{(material A)}$$

$$x_2 + 2x_3 + 3x_4 \leq 90 \quad \text{(material B)}$$

$$\max\ 20x_1 + 12x_2 + 40x_3 + 25x_4 \quad \text{(profit)}$$

$$x_1 + x_2 + x_3 + x_4 \leq 50 \quad \text{(manpower)}$$

Resource Allocation Problem

https://realpython.com/linear-programming-python/

Code - PuLP

- PuLP is an LP modeler written in Python

Optimization with PuLP

You can begin learning Python and using PuLP by looking at the content below. We recommend that you read The Optimisation Process, Optimisation Concepts, and the Introduction to Python before beginning the case-studies. For instructions for the installation of PuLP see Installing PuLP at Home.

https://coin-or.github.io/pulp/index.html
https://github.com/coin-or/pulp

Steps to LP

1. Understand the problem, describe the objective
2. Define the decision variables, write the objective function
3. Describe the constraints, write the constraints in terms of the decision variables
4. Coding and solving

http://www.mit.edu/~hlb/MATH318/linearprogrammingsteps.html

Example 2

- Suppose you are in charge of the diet plan for high school lunch
- Your job is to make sure that the students get the right balance of nutrition from the chosen food

A	B	C	D	E	F	G	H	I	J	K	L	M	N
Foods	**Price/Serving**	**Serving Size**	**Calories**	**Cholesterol (mg)**	**Total_Fat (g)**	**Sodium (mg)**	**Carbohydrates (g)**	**Dietary_Fiber (g)**	**Protein (g)**	**Vit_A (IU)**	**Vit_C (IU)**	**Calcium (mg)**	**Iron (mg)**
Frozen Broccoli	$0.48	10 Oz Pkg	73.8	0	0.8	68.2	13.6	8.5	8	5867.4	160.2	159	2.3
Frozen Corn	$0.54	1/2 Cup	72.2	0	0.6	2.5	17.1	2	2.5	106.6	5.2	3.3	0.3
Raw Lettuce Iceberg	$0.06	1 Leaf	2.6	0	0	1.8	0.4	0.3	0.2	66	0.8	3.8	0.1
Baked Potatoes	$0.18	1/2 Cup	171.5	0	0.2	15.2	39.9	3.2	3.7	0	15.6	22.7	4.3
Tofu	$0.93	1/4 block	88.2	0	5.5	8.1	2.2	1.4	9.4	98.6	0.1	121.8	6.2
Roasted Chicken	$2.52	1 lb chicken	277.4	129.9	10.8	125.6	0	0	42.2	77.4	0	21.9	1.8
Spaghetti W/ Sauce	$2.34	1 1/2 Cup	358.2	0	12.3	1237.1	58.3	11.6	8.2	3055.2	27.9	80.2	2.3
Raw Apple	$0.72	1 Fruit,3/Lb,Wo/Rf	81.4	0	0.5	0	21	3.7	0.3	73.1	7.9	9.7	0.2
Banana	$0.45	1 Fruit,Wo/Skn&Seeds	104.9	0	0.5	1.1	26.7	2.7	1.2	92.3	10.4	6.8	0.4
Wheat Bread	$0.15	1 Sl	65	0	1	134.5	12.4	1.3	2.2	0	0	10.8	0.7
White Bread	$0.18	1 Sl	65	0	1	132.5	11.8	1.1	2.3	0	0	26.2	0.8
Oatmeal Cookies	$0.27	1 Cookie	81	0	3.3	68.9	12.4	0.6	1.1	2.9	0.1	6.7	0.5
Apple Pie	$0.48	1 Oz	67.2	0	3.1	75.4	9.6	0.5	0.5	35.2	0.9	3.1	0.1
Scrambled Eggs	$0.33	1 Egg	99.6	211.2	7.3	168	1.3	0	6.7	409.2	0.1	42.6	0.7
Turkey Bologna	$0.45	1 Oz	56.4	28.1	4.3	248.9	0.3	0	3.9	0	0	23.8	0.4
Beef Frankfurter	$0.81	1 Frankfurter	141.8	27.4	12.8	461.7	0.8	0	5.4	0	10.8	9	0.6
Chocolate Chip Cookies	$0.09	1 Cookie	78.1	5.1	4.5	57.8	9.3	0	0.9	101.8	0	6.2	0.4
		Minimum daily intake	800	30	20	800	130	60	100	1000	400	700	10
		Maximum daily intake	1300	240	50	2000	200	125	150	10000	5000	1500	40

$$Minimize: \sum C_i.f_i$$

$$s.t.\ calorie_{lower} < \sum Cal_i.f_i < calorie_{upper}$$

$$s.t.\ protein_{lower} < \sum Pro_i.f_i < protein_{upper}$$

https://towardsdatascience.com/linear-programming-and-discrete-optimization-with-python-using-pulp-449f3c5f6e99

Thanks