

— Линейное программирования

Отчет по лабораторной работе №1

Работу выполнили:

Обиджанов Алишер
Какзаков Андрей
Кузнецов Павел

Преподаватель:

Свинцов М.В.

1 Постановка задачи

1. Реализуйте возможность ввода данных из файла в формате JSON. Рекомендуемая структура JSON указана ниже.
2. При необходимости добавьте балансирующие переменные для перехода от общей постановки к канонической форме задачи линейного программирования.
3. Реализуйте симплекс-метод для решения задачи.
4. Предусмотрите, что задача как может не иметь решений вообще, так и иметь бесконечное количество решений

2 Реализация

2.1 Подключаем библиотеки

```
import numpy as np
from json import loads
```

2.2 Подготавливаем пример в формате json

```
example = """{"f": [1, 2, 3],
  "goal": "max",
  "constraints": [{"coefs": [1, 0, 0],
    "type": "eq",
    "b": 1},
    {"coefs": [1, 1, 0],
    "type": "gte",
    "b": 2},
    {"coefs": [1, 1, 1],
    "type": "lte",
    "b": 3}]}"""
```

2.3 Парсим json файл

Преобразует json в матрицы f , A и b , которые затем возвращает. Если задача имеет цель "min" то целевая функция домножается на -1. Коэффициенты у всех ограничений в виде неравенств вида "больше" инвертируются, тем самым превращаясь в ограничения вида "меньше". Когда встречается равенство, оно делится на два неравенства, одно с неизменённым знаком, другое - с изменённым.

```

def parse_problem(json: str):
    parsed = loads(json)
    f = np.array(parsed['f'])
    if parsed['goal'] == "min":
        f *= -1
    A = []
    b = []
    for constraint in parsed['constraints']:
        A.append(np.array(constraint['coefs']))
        b.append(constraint['b'])
        if constraint['type'] == "gte":
            A[-1] *= -1
            b[-1] *= -1
        if constraint['type'] == "eq":
            A.append(np.array(constraint['coefs']) * -1)
            b.append(-constraint['b'])
    A = np.array(A)
    A = np.hstack((A, np.eye(A.shape[0])))
    b = np.array(b)
    b = b.reshape((b.shape[0], 1))
    return f, A, b

```

2.4 Симплекс-метод

Сначала создается таблица `tableau`, которая объединяет матрицы `A` и `b`. Если в таблице есть отрицательные элементы в столбце `b`, то выбирается строка `i` с минимальным значением `b` и столбец `l` с минимальным значением в этой строке. Затем находятся отношения между элементами строки `i` и столбцом `l` и выбирается строка `r` с наименьшим отношением. Далее выполняется шаг симплекс-метода для выбранных строк и столбцов.

2.4.1 Шаг симплекс-метода

```

def simplex_step(tableau, r, l):
    for i in range(tableau.shape[0]):
        if i == r:
            tableau[i] /= tableau[i, l]
            continue
        tableau[i] -= tableau[r] * tableau[i, l] / tableau[r, l]

```

2.4.2 Симплекс-метод

```
def simplex(f, A, b):
    tableau = np.hstack((b, A))
    tableau = np.vstack((tableau, np.hstack((np.zeros((1,)), f, np.zeros((A.shape[0]))))))
    print(tableau)
    if b[b < 0].any():
        i = tableau[:-1, 0].argmin()
        l = tableau[i, 1:].argmin()
        if tableau[i, l] >= 0:
            raise Exception("No solution.")
        ratios = tableau[:-1, 0] / tableau[i, l]
        r = ratios.argmin()
        simplex_step(tableau, r, l)
        print(tableau)
    s = tableau[-1].argmax()

    last_max = -1
    while s > 0:
        print(tableau[:-1, s])
        temp = tableau[:-1, s]
        temp[temp == 0] = -1
        ratios = tableau[:-1, 0] / temp
        j = ratios.argmin()
        simplex_step(tableau, j, s)
        print(tableau)
        s = tableau[-1].argmax()
        if tableau[-1, s] == last_max:
            break
        last_max = tableau[-1, s]
    return tableau[-1, 0]
```

3 Результат

```
print(simplex(*parse_problem(example)))
```

Output:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 2 & 0 & 1 & 0 \\ 0 & -2 & 1 & 1 & -3 & 0 & 0 & 1 \\ 0 & 1 & 2 & 3 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$(0 \ 0 \ 0 \ 1)$$

$$\begin{pmatrix} -1 & -1 & -0 & 1 & -1 & -0 & -0 & -0 \\ -1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & -1 & 0 & 1 & 0 & 1 & 0 \\ 1 & -1 & 1 & 0 & -2 & 0 & 0 & 1 \\ 3 & 4 & 2 & 0 & 3 & 0 & 0 & 0 \end{pmatrix}$$

$$(-1 \ -1 \ 0 \ -1)$$

$$\begin{pmatrix} -2 & 0 & -1 & 1 & 1 & -0 & -0 & -1 \\ -2 & 0 & -1 & 0 & 2 & 1 & 0 & -1 \\ -2 & 0 & -2 & 0 & 3 & 0 & 1 & -1 \\ -1 & 1 & -1 & -0 & 2 & -0 & -0 & -1 \\ 7 & 0 & 6 & 0 & -5 & 0 & 0 & 4 \end{pmatrix}$$

7.0