Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по лабораторной работе №6
Задача квадратичного программирования.

Выполнил:

студент гр. 953505

Басенко К. А.

Руководитель:

доцент

Алёхина А. Э.

Минск 2022

Постановка задачи

Задача квадратичного программирования с n переменными и m ограничениями можно сформулировать следующим образом.

Дано:

- вещественный n-мерный вектор c,
- $n \times n$-мерная вещественная симметричная матрица D,
- $m \times m$-мерная вещественная матрица A
- вещественный m-мерный вектор b,

Целью задачи квадратичного программирования является поиск n-мерного вектора x, который минимизирует $c'x + \frac{1}{2}x'Dx$, при $Ax = b, x \geq 0$, где x' обозначает транспонированный вектор.

Краткие теоретические сведения

Задача квадратичного программирования в канонической форме состоит в минимизации квадратичной функции

$$f(x) := c'x + 1/2x'Dx \to \min,$$

при линейных ограничениях

$$Ax = b, \quad x \geq 0.$$

Здесь $A = (A_j, j \in J)$ – $m \times m$-матрица со столбцами A_j, $j \in J = \{1, 2, \ldots, n\};$ x и c – n-векторы, b – m-вектор, D – симметричная ($D=D'$) положительно полуопределенная ($D \geq 0$) $n \times n$-матрица.

Пусть известны параметры c, b, A, D, определяющие задачу. Для реализации итерации метода необходимо знать текущий правильный опорный план $\{x, J_{on}, J_*\}$.

Опишем итерацию метода по шагам.

Шаг 1. Сформируем вектор $c(x) = c + Dx$ и подсчитаем вектор потенциалов

$$u'(x) = -c_{on}(x)A_{on}^{-1}, \text{ где } c_{on}(x) = (c_j(x), j \in J_{on}), \; A_{on} = (A_j, j \in J_{on})$$

и оценки

$$\Delta_j = u'(x)A_j + c_j(x), \; j \in J_H = J \setminus J_*.$$

Шаг 2. Если Δ_j ³ 0, $j \in J_H$, то STOP: вектор x – решение задачи. В противном случае перейдем к шагу 3.

Шаг 3. Выберем индекс $j_0 \in J_H$, для которого $\Delta_{j_0} < 0$.

Шаг 4. По заданным множеству J_* и индексу j_0 сформируем систему и найдем ее решение (l_*, y), $l_* = (l_j, j \in J_*)$, $y \in R^m$. *Положим* $\delta := D'_{*j_0} l_* + A'_{j_0} y + d_{j_0 j_0}$.

Шаг 5. Вычислим

$\Theta_j = -x_j / l_j$ при $l_j < 0$, $\Theta_j = \infty$ при $l_j \geq 0, j \in J_*$;

$\Theta_{j_0} = |\Delta_{j_0}/\delta|$ при $\delta \neq 0$, $\Theta_{j_0} = \infty$ при $\delta = 0$.

Найдем $\Theta_0 = min\ \Theta_j$, $j \in J_* \cup j_0$. Если $\Theta_0 = \infty$, то STOP: задача не имеет решения в силу неограниченности снизу целевой функции на множестве планов. В противном случае выберем индекс $j_* \in J_* \cup j_0$, для которого $\Theta_0 = \Theta_{j*}$, и перейдем к шагу 6.

Шаг 6. Построим новый план $\bar{x} = \left(\bar{x}_j, j \in J\right)$ по правилам:

$$\bar{x}_j = x_j + \Theta_0 l_j, j \in J_*; \bar{x}_{j_0} = x_{j_0} + \Theta_0, \bar{x}_j = 0, j \in J_H \setminus j_0.$$

Шаг 7. Предполагается, что множество индексов J_{on} имеет вид $J_{on} = \{j_1, j_2, ..., j_s, ..., j_m\}$. Возможны случаи:

а) $j_* = j_0$;

б) $j_* \in J_* \setminus J_{on}$;

в) $j_* = j_s \in J_{on}$ и существует такой индекс $j_+ \in J_* \setminus J_{on}$, что $e_s' A_{on}^{-1} A_{j+} \neq 0$;

г) $j_* = j_s \in J_{on}$ и $e_s' A_{on}^{-1} A_j = 0, j \in J_* \setminus J_{on}$, либо $J_* = J_{on}$.

В случае а полагаем

$$\bar{J}_{on} = J_{on},\ \bar{J}_* = J_* \cup j_0$$

и переходим к новой итерации, т.е. к шагу 1, используя найденные данные $\bar{x}, \bar{J}_{on}, \bar{J}_*$.

В случае б полагаем

$$\bar{J}_{on} = J_{on},\ \bar{J}_* = J_* \setminus j_*,\quad \bar{j}_0 = j_0,\ \bar{\Delta}_{j_0} = \Delta_{j_0} + \Theta_0 \delta$$

и переходим к шагу 4, используя далее данные $\bar{x}, \bar{J}_{on}, \bar{J}_*, \bar{j}_0, \bar{\Delta}_{\bar{j}_0}$ вместо $x, J_{on}, J_*, j_0, \Delta_{j_0}$.

В случае в полагаем

$$\bar{J}_{on} = (J_{on} \setminus j_*) \cup j_+,\ \bar{J}_* = J_* \setminus j_*,\ \bar{j}_0 = j_0,\ \bar{\Delta}_{j_0} = \Delta_{j_0} + \Theta_0 \delta$$

и переходим к шагу 4, используя обновленные данные $\bar{x}, \bar{J}_{on}, \bar{J}_*, \bar{j}_0, \bar{\Delta}_{\bar{j}_0}$ вместо $x, J_{on}, J_*, j_0, \Delta_{j_0}$.

В случае г полагаем

$$\bar{J}_{on} = (J_{on} \setminus j_*) \cup j_0, \ \bar{J}_* = (J_* \setminus j_*) \cup j_0$$

и переходим к новой итерации (т.е. к шагу 1), используя новые данные $\bar{x}, \bar{J}_{on}, \bar{J}_*$.

Программная реализация

Решение задачи квадратичного программирования методом опорных планов реализована в функции *quadratic_programming_problem* на вход в которую идут: параметры A, b, c, D, текущий базисный опорный план $\{x, J_{оп}, J_*\}$.

```
def quadratic_programming_problem(A, b, c, D, x, J, J_op):
```

Сформируем вектор $c(x) = c + Dx$:

```
cx = c + np.matmul(D, x)

c_op = cx[J_op]
A_op = A[:, J_op]
A_op_inv = np.linalg.inv(A_op)
```

Подсчитаем вектор потенциалов:

```
ux = -np.matmul(c_op, A_op_inv)
```

Подсчитаем вектор оценок:

```
deltas = np.matmul(ux, A) + cx
deltas = np.array([round(_x, rounding) for _x in deltas])
```

Проверяем условие оптимальности:

```
if all(deltas >= 0):
    print('План оптимальный')
    print(f'x\' = {[round(_x, rounding) for _x in x]}')
    return
```

Запоминаем индекс, элемент которого не удовлетворил условие оптимальности:

```
j0 = np.argmin(deltas)
```

Составим систему и найдем l^*, y:

$$D_* l_* + A'_* y + D_{*j_0} = 0,$$
$$A_* l_* + A_{j_0} = 0$$

```
DJ = D[J][:, J]
AJ = A[:, J]

Aj0 = A[:, j0]
Dj0 = D[j0][J]

l_len = len(J)
y_len = m
syst = np.zeros((l_len + y_len, l_len + y_len))

syst[:l_len, :l_len] = DJ
syst[l_len:, :l_len] = AJ
syst[:l_len, l_len:] = AJ.T

syst_b = np.zeros(l_len + y_len)
syst_b[: l_len] = -Dj0
syst_b[l_len: ] = -Aj0
```

Решим ее:

```
res = np.linalg.solve(syst, syst_b)

lJ = res[:l_len]
y = res[-y_len:]

l = np.zeros(n)
l[J] = lJ
```

Вычислим

$$\delta := D'_{*j_0} l_* + A'_{j_0} y + d_{j_0 j_0}$$

$\Theta_j = -x_j / l_j$ при $l_j < 0$, $\Theta_j = \infty$ при $l_j \geq 0, j \in J_*$;

$\Theta_{j_0} = |\Delta_{j_0} / \delta|$ при $\delta \neq 0$, $\Theta_{j_0} = \infty$ при $\delta = 0$:

```python
delta = np.matmul(Dj0, lJ) + np.matmul(Aj0, y) + D[j0][j0]
theta_j0 = abs(deltas[j0] / delta) if delta != 0 else np.inf

thetas = [(-x[j] / l[j] if l[j] < 0 else np.inf) for j in J]
```

Проверяем ограниченность снизу целевой функции:

```python
theta0 = min(thetas)
if theta0 == np.inf:
    print('Задача не имеет решения в силу неограниченности снизу целевой функции на множестве планов')
    return
```

Выберем индекс $j_* \in J_* \bigcup j_0$, для которого $\Theta_0 = \Theta_{j*}$:

```python
jJ = np.argmin(thetas)
if thetas[jJ] >= theta_j0:
    theta0 = theta_j0
    jJ = j0
else:
    theta0 = thetas[jJ]
    jJ = J[jJ]
```

Строим новый план:

```python
x[J] += theta0 * lJ
x[j0] += theta0

for j in range(n):
    if j not in J and j != j0:
        x[j] = 0
```

Обновляем множество индексов:

```python
if j0 == jJ:
    J = np.append(J, [j0])
elif (jJ in J) and (jJ not in J_op):
    J = [j for j in J if j != jJ]
    deltas[j0] += theta0 * delta
    continue
else:
    s = np.where(J_op == jJ)
    es = np.eye(m)[s]
    j_plus = np.array(set(J) - set(J_op))

    if j_plus and any(es.dot(A_op_inv).dot(A[:, j_plus]) != 0):
        J = J[J != jJ]
        J_op = np.append(J_op[J_op != jJ], j_plus)
        deltas[j0] += theta0 * delta
        continue
    else:
        J = np.append(J[J != jJ], j0)
        J_op = np.append(J_op[J_op != jJ], j0)
```

Далее повторяем вычисления с новым множеством индексов и планом.

Тестовые примеры

Данные для тестирования:

```
test_cases = [
    [
        A := [
            [1, 0, 2, 1],
            [0, 1, -1, 2]
        ],
        b := [2, 3],
        c := [-8, -6, -4, -6],
        D := [
            [2, 1, 1, 0],
            [1, 1, 0, 0],
            [1, 0, 1, 0],
            [0, 0, 0, 0]
        ],
        x := [2, 3, 0, 0],
        J := [0, 1],
        J_op := J[:],
        res := [1.7, 2.4, 0, 0.3],
    ],
```

```
    [
        A := [
            [6, 6, 0],
            [3, 0, 1]
        ],
        b := [3, 1],
        c := [-1, 0, 0],
        D := [
            [4, -2, 0],
            [-2, 4, 0],
            [0, 0, 1],
        ],
        x := [0, 0.5, 1],
        J := [1, 2],
        J_op := J[:],
        res := [1/3., 1/6., 0],
    ],
```

```
    [
        A := [
            [2, 1, 0],
            [5, 0, 7]
        ],
        b := [6./5, 2],
        c := [-1, -1, 0],
        D := [
            [2, 1, 0],
            [1, 2, 0],
            [0, 0, 2],
        ],
        x := [0, 6./5, 2./7],
        J := [1, 2],
        J_op := J[:],
        res := [2./5, 2./5, 0],
    ],
```

```
    [
        A := [
            [3./2, 0, 3],
            [3, 1, 0]
        ],
        b := [1, 1],
        c := [-1, 0, 0],
        D := [
            [4, 0, -2],
            [0, 1, 0],
            [-2, 0, 4],
        ],
        x := [0, 1, 1./3],
        J := [1, 2],
        J_op := J[:],
        res := [1./3, 0, 1./6],
    ]
]
```

Выполняется метод с предоставленными данными, выводится полученный результат, выводится ожидаемый результат:

```
for *data, res in test_cases:
    quadratic_programming_problem(*data)
    print(f'Правильный ответ:\nx\' = {[round(_x, rounding) for _x in res]}\n\n')
```

Вывод, соответствующий предоставленным входным данным:

```
План оптимальный
x' = [1.7, 2.4, 0.0, 0.3]
Правильный ответ:
x' = [1.7, 2.4, 0, 0.3]
```

```
План оптимальный
x' = [0.33333, 0.16667, 0.0]
Правильный ответ:
x' = [0.33333, 0.16667, 0]
```

```
План оптимальный
x' = [0.4, 0.4, 0.0]
Правильный ответ:
x' = [0.4, 0.4, 0]
```

```
План оптимальный
x' = [0.33333, 0.0, 0.16667]
Правильный ответ:
x' = [0.33333, 0, 0.16667]
```