NANCHANG UNIVERSITY

最优化理论实验报告

EXPERIMENTAL REPORT

题　　　目: 线性规划

院　　　系: 信息工程学院

专业班级: 计算机技术 2021 级

名　　　字: 方绍雷

日　　　期: 2022 年 1 月 19 日

1 题目描述

在给定一个定义在 n 维空间中的凸函数 $f(x)$, 在空间中任取 m 个点 $\{x_i\}_{i=1}^m$，另记

$$p_i = \frac{1}{2}\nabla f(x_i), i = 1, ..., m \tag{1}$$

再为每个点定义一个权重 $w_i, i = 1, ..., m$, 对任意一个 w_i 需要满足下列 $m-1$ 个约束：

$$\forall j \neq i, ||x_i - p_i|| - w_i \leq ||x_i - p_j|| - w_j, j = 1, ..., m \tag{2}$$

若假定已知 $f(x)$,$\{x_i\}_{i=1}^m$,$\{p_i\}_{i=1}^m$, 求解 m 维的线性规划问题

$$min \sum_{i=0}^{m} w_i \tag{3}$$

$$s.t. \forall, j = 1, ..., m, \forall j \neq i, ||x_i - p_i|| - w_i \leq ||x_i - p_j|| - w_j$$

一共有 $m(m-1)$ 个约束条件。
要求测试至少两个 $f(x)$, $10 \leq m \leq 1000$，不要求 n 很高，2/3 维度就可以。

2 问题分析

可以分析要解决此问题，就需要把上述的非标准形式转化为标准的线性规划表述形式，在利用算法可以求解出此问题。为了问题的求解，假定 w_i 都是大于零的数。标准化公式 (2):

$$w_j - w_i \leq ||x_i - p_i||_2 - ||x_i - p_i||_2, \forall j \neq i, j \in \{1, 2, ..., m\} \tag{4}$$

假设 $i = k$, 其中满足约束 $k \in \{1, 2, ..., m\}$ $k \neq j$ 将上述 (4) 公式展开得到：

$$\begin{aligned} w_1 - w_k &\leq ||x_k - p_1|| - ||x_k - p_k|| \\ w_2 - w_k &\leq ||x_k - p_2|| - ||x_k - p_k|| \\ &.... \\ w_m - w_k &\leq ||x_k - p_m|| - ||x_k - p_k|| \end{aligned} \tag{5}$$

一共得到 m-1 项式子，将公式（5）进行连加得到：

$$(1-m)w_k + \sum_{j\neq k} w_j \leq (1-m)||x_k - p_k|| + \sum_{j\neq k} ||x_k - p_j||$$

则可以对整体式子构造 m 阶方阵 A 和 m 维向量 b 得到：

$$A = \begin{bmatrix} 1-m & 1 & \cdots & 1 \\ 1 & 1-m & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1-m \end{bmatrix}$$

其中 $b = (b_1, b_2, ..., b_m)^T$，

$$b_k = (1-m)||x_k - p_k|| + \sum_{j\neq k} ||x_k - p_j||$$

从而得到问题的标准化形式：

$$Aw \leq b$$

得到了上述标准的线性规划表述的形式，可以通过求极点和极方向来求解这类问题，基于极点和方向，可以有穷举法，单纯性法等等。

3　代码实现

对于代码的实现，我并没有自己去实现解决此问题的方法，而是利用一些现有的工具去解决。导入必要的库函数：

```
import numpy as np
from scipy import optimize
```

其他代码：

```
def f(x):
    """
    定义给定的凸函数：定义为f(x) = 0.1 x_1 ^ 2 + x_2^2,
        此函数返回目标函数的值
    :param x: 二维向量
```

```
    :return:
    """
    return 0.1 * x[0] * x[0] + x[1] * x[1]
def grad_f(x):
    '''
    给定凸函数的梯度（解析解): grad(x) = (0.2 * x[0], 2 * x[1])
    :param x: 2D实向量
    :return: 2D实向量
    '''
    y = np.empty_like(x)
    y[0] = 0.2 * x[0]
    y[1] = 2 * x[1]
    return y
'''
随机选取m个点，这里选取m=15
'''
m = 15
X = np.random.normal(size=(m, 2))
P = grad_f(X)
'''
标准化参数
'''
c = np.ones((m, ))
A = np.ones((m, m)) - np.identity(m) * m
'''
求解bn
'''
tmp = []
for i in range(m):
t = np.sqrt(np.sum(np.square(X[i:i + 1] - P), axis=-1))
t[i] *= (1 - m)
tmp.append(np.sum(t))
b_n = np.array(tmp)
res = optimize.linprog(c, A_ub=A, b_ub=b_n, bounds=[0, None],
```

```
    method='revised simplex')
print(res)
print(b_n)
```

4 结果展示与分析

按照上述思想和代码所实现的那样，我选取了两个测试案例，如下：

1. 令 $f(x) = 0.1 * x_1^2 + x_2^2$, 且 $m = 15$。首先得到 m 个二维随机点为：

```
[[ 0.86352056 -0.30690865]
 [ 0.33123031  0.23255507]
 [ 0.38145427 -0.54738266]
 [-0.71263718 -0.6313543 ]
 [ 0.36301042  0.49599476]
 [ 0.33865445 -0.22429897]
 [ 0.07664525 -0.23054585]
 [-0.42751702  0.61303978]
 [ 0.45484447  0.30407656]
 [-1.45585134  1.2708531 ]
 [ 0.40882174 -2.12649226]
 [ 0.3189274   1.86105974]
 [ 0.01085958  0.77453807]
 [-0.82047378 -1.6285548 ]
 [-3.01278359  0.58547593]]
```

Fig. 1. 得到 m 个随机点的坐标

在上述中 m 个随机点的前提下，运行得到最终的结果为：

```
     con: array([], dtype=float64)
     fun: 0.11670748749659825
 message: 'Optimization terminated successfully.'
     nit: 5
   slack: array([ 2.33065365e+00, -1.38777878e-17,  1.28624362e-01,  7.90805066e-01,
        0.00000000e+00,  6.51169450e-02,  5.01382171e-01,  3.91321356e-01,
        5.55111512e-17,  3.88716426e-01,  2.34974159e-01,  0.00000000e+00,
       -2.08166817e-17,  6.57197545e-01,  6.72719520e-01])
  status: 0
 success: True
       x: array([0.        , 0.01249758, 0.        , 0.        , 0.03039917,
       0.        , 0.        , 0.        , 0.03243431, 0.        ,
       0.        , 0.03516616, 0.00621026, 0.        , 0.        ])
```

Fig. 2. 最终的运行结果

2. 令 $f(x) = 0.1 * x_1^2 + 0.2 * x_2^2 + 0.3 * x_3^2$, 且 $m = 12$。首先得到 m 个二维随机点为：

```
[[-0.12605777  1.49399801]
 [ 0.40053279 -0.56125443]
 [-0.19615697  0.38973894]
 [-0.60518232  0.43710936]
 [ 0.78629031 -0.11947598]
 [ 0.79161578  0.93034636]
 [-0.79996777  1.02410437]
 [ 1.13780585 -0.71348274]
 [-0.47564886 -1.747495  ]
 [ 0.73531758 -0.65133613]
 [ 0.87858882 -0.4886015 ]
 [ 1.42454179  1.31978031]]
```

Fig. 3. 得到 m 个随机点的坐标

在上述中 m 个随机点的前提下，运行得到最终的结果为：

```
     con: array([], dtype=float64)
     fun: 0.08284869630948774
 message: 'Optimization terminated successfully.'
     nit: 4
   slack: array([ 3.21937544e+00,  3.48072609e+00,  2.82798127e+00, -2.77555756e-17,
        4.95646576e-02,  0.00000000e+00,  0.00000000e+00,  9.58636970e-02,
        2.43036320e-01,  1.32745397e-01,  9.59447292e-02,  0.00000000e+00])
  status: 0
 success: True
       x: array([0.        , 0.        , 0.        , 0.01552759, 0.        ,
       0.02111857, 0.02508082, 0.        , 0.        , 0.        ,
       0.        , 0.02112171])
```

Fig. 4. 最终的运行结果