

使用python求解二次规划的问题

今天小编就为大家分享一篇使用python求解二次规划的问题，具有很好的参考价值，希望对大家有所帮助。一起跟随小编过来看看吧

Python中支持Convex Optimization（凸规划）的模块为CVXOPT,其安装方式为：

```
pip install cvxopt
```

一、数学基础

二次型

二次型（quadratic form）：n个变量的二次多项式称为二次型，即在一个多项式中，未知数的个数为任意多个，但每一项的次数都为2的多项式。其基本形式如下

$$f(x_1, x_2, \dots, x_n) = a_{11}x_1^2 + a_{22}x_2^2 + \dots + a_{nn}x_n^2 \\ + 2a_{12}x_1x_2 + 2a_{13}x_1x_3 + \dots + 2a_{1n}x_1x_n \\ + 2a_{23}x_2x_3 + \dots + 2a_{n-1,n}x_{n-1}x_n$$

亦可写作， $f(\vec{x}) = \vec{x}^T \mathbf{A} \vec{x}$ ，称作二次型的矩阵表示，其中A是对称矩阵。仿照如下的定义，我们可以直接在其基本形式和矩阵表示之间相互转化。

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

2.正定矩阵

设A是n阶实对称矩阵，如果对任意一非零实向量X，都使二次型 $f(\vec{x}) = \vec{x}^T \mathbf{A} \vec{x} > 0$ 成立，则称f(X)为正定二次型，矩阵A称为正定矩阵(Positive Definite)，A为正定矩阵。

相应的，如果对任意一非零实向量X，都使二次型 $f(\vec{x}) = \vec{x}^T \mathbf{A} \vec{x} \geq 0$ 成立，则称f(X)为半正定二次型，A为半正定矩阵。

3.二次规划问题

二次规划是指，带有二次型目标函数和约束条件的最优化问题。其标准形式如下：

$$\min_x \quad \frac{1}{2}x^T Px + q^T x \\ \text{subject to} \quad Gx \preceq h \\ \quad \quad \quad Ax = b$$

即在 $Gx \preceq h$ 和 $Ax=b$ 的约束下，最小化目标函数。其中，当P是正定矩阵时，目标函数存在全局唯一最优解；P是半正定矩阵时，目标函数是凸函数，存在全局最优解（不唯一）；P是不定矩阵时，目标函数非凸，存在多个局部最小值和稳定点，为np

难问题。（本篇博客中我们不考虑非正定情况）。

二、python程序求解

工具包：Cvxopt python 凸优化包

函数原型：Cvxopt.solvers.qp(P,q,G,h,A,b)

P,q,G,h,A,b的含义参见上面的二次规划问题标准形式。

编程求解思路：

- 1.对于一个给定的二次规划问题，先转换为标准形式（参见数学基础中所讲的二次型二中形式转换）
- 2.对照标准形势，构建出矩阵P,q,G,h,A,b
- 3.调用result=Cvxopt.solvers.qp(P,q,G,h,A,b)求解
- 4.print (result) 查看结果，其中result是一个字典，我们可直接获得其某个属性，e.g. print(result['x'])

下面我们来看一个例子

例：

$$\begin{aligned} \min \quad & 2x_1^2 + x_2^2 + x_1x_2 + x_1 + x_2 \\ \text{s. t.} \quad & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_1 + x_2 = 1 \end{aligned}$$

标准形式：

$$\begin{aligned} \min \quad & 2x_1^2 + x_2^2 + x_1x_2 + x_1 + x_2 \\ \text{s. t.} \quad & -x_1 \leq 0 \\ & -x_2 \leq 0 \\ & x_1 + x_2 = 1 \end{aligned}$$

$$\text{则 } P = \begin{bmatrix} 4 & 1 \\ 1 & 2 \end{bmatrix} \quad q = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad G = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \quad h = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad A = [1, 1] \quad b = 1$$

https://blog.csdn.net/goodxin_ie

```
import pprint
from cvxopt import matrix, solvers
P = matrix([[4.0,1.0],[1.0,2.0]])
q = matrix([1.0,1.0])
G = matrix([[-1.0,0.0],[0.0,-1.0]])
h = matrix([0.0,0.0])
A = matrix([1.0,1.0],[1,2])#原型为cvxopt.matrix(array,dims)，等价于A = matrix([[1.0],[1.0]])
b = matrix([1.0])
result = solvers.qp(P,q,G,h,A,b)
```

```
print('x\n',result['x'])
```

运行结果：

```
PS C:\Users\goodxin\Desktop\临时> python .\sol.py
      pcost      dcost      gap      pres      dres
0:  1.8889e+00  7.7778e-01  1e+00  3e-16  2e+00
1:  1.8769e+00  1.8320e+00  4e-02  2e-16  6e-02
2:  1.8750e+00  1.8739e+00  1e-03  2e-16  5e-04
3:  1.8750e+00  1.8750e+00  1e-05  6e-17  5e-06
4:  1.8750e+00  1.8750e+00  1e-07  2e-16  5e-08
Optimal solution found.
x
[ 2.50e-01]
[ 7.50e-01]
```

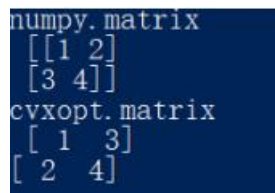
https://blog.csdn.net/goodxin_ie

注意事项：

cvxopt.matrix与numpy.matrix的排列顺序不同，其中cvxopt.matrix是列优先，numpy.matrix是行优先。具体可见下面实例

```
import numpy as np
from cvxopt import matrix
a = np.matrix([[1,2],[3,4]])
b = matrix([[1,2],[3,4]])
print('numpy.matrix',a)
print('cvxopt.matrix',b)
```

运行结果：



```
numpy.matrix
[[1 2]
 [3 4]]
cvxopt.matrix
[ 1 3]
[ 2 4]
```

以上这篇使用python求解二次规划的问题就是小编分享给大家的全部内容了，希望能给大家一个参考，也希望大家多多支持我们。