非线性规划和整数规划模型

非线性规划概念和理论

记$x=[x_1,x_2,\dots,x_n]^T$为一n维向量，

$$
\begin{gathered}
max(or\ min)\ f(x),\\
s.t.\begin{cases} g_i(x)\le 0, & i=1,2,\dots,p,\\ h_j(x)=0, & j=1,2,\dots,q. \end{cases}
\end{gathered}
$$

其中$f_i(x),g_j(x)$中至少有一个为非线性函数.

称$K=\{x\in R^n\ |g_i(x)\le 0,i=1,\dots,p;h_i(x)=0,j=1,\dots,q\}$为其约束集或者可行域，
若$x^*\in K$，且$\forall x\in K$，都有$f(x^*)\le f(x)$，则称x^*为其全局最优解，对应的函数f的值为全局最优值；
若$x^*\in K$，且存在x^*的邻域$N_\delta(x^*)$，$\forall x\in N_\delta(x^*)\cap K$，都有$f(x^*)\le f(x)$，则称$x^*$为其局部最优解，对应的函数f的值为局部最优值.

若$f(x),g_i(x)$为凸函数，$h_j(x)$为线性函数，则称之为凸规划.
凸规划的局部最优解必是其全局最优解.

二次规划模型

1.模型形式

目标函数为决策向量X的二次函数，

$$
\begin{gathered}
max(or\ min)\ \sum_{i=1}^n\sum_{j=1}^n c_{ij}x_ix_j+\sum_{i=1}^n d_ix_i\\
s.t.\begin{cases} \sum_{j=1}^n a_{ij}x_j\le (or =,\ge)0, & i=1,2,\dots,m,\\ x_i\ge 0, & i=1,2,\dots,n. \end{cases}
\end{gathered}
$$

其中$c_{ij}=c_{ji},\quad i,j=1,2,\dots,n$，
即$H=(c_{ij})_{n\times n}$为对称矩阵，当H正定时，目标函数最小化时，模型为凸二次规划，其局部最优解为其全局最优解.

2.求解方法

求解下列二次规划模型：

$$
\begin{gathered}
max - x_1^2 - 0.3x_1x_2 - 2x_2^2 + 98x_1 + 277x_2, \\
s.t. \begin{cases} x_1 + x_2 \leq 100, \\ x_1 - 2x_2 \leq 0, \\ x_1, x_2 \geq 0. \end{cases}
\end{gathered}
$$

解法：

```
#目标函数为凸函数，使用cvxpy库求解
import cvxpy as cp
import numpy as np

c2 = np.array([[-1, -0.15], [-0.15, -2]])
c1 = np.array([98, 277])
a = np.array([[1, 1], [1, -2]])
b = np.array([100, 0])
x = cp.Variable(2, pos = True)
obj = cp.Maximize(cp.quad_form(x, c2) + c1 @ x)
con = [a @ x <= b]
prob = cp.Problem(obj, con)
prob.solve(solver = 'CVXOPT')
print('最优解为', np.round(x.value, 4))
print('最优值为', round(prob.value, 4))
```

一般非线性规划的求解

求解下列一般线性规划：

$$
\begin{gathered}
min\ f(x) = x_1^2 + x_2^2 + x_3^2 = 8 \\
s.t. \begin{cases} x_1^2 - x_1 + x_3^2 \geq 0, \\ x_1 + x_2^2 + x_3^2 \leq 20, \\ -x_1 - x_2^2 + 2 = 0, \\ x_2 + 2x_3^2 = 3, \\ x_1, x_2, x_3 \geq 0. \end{cases}
\end{gathered}
$$

解法：

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

obj = lambda x: sum(x ** 2) + 8



def constr1(x):
```

```python
        x1, x2, x3 = x
        return [x1 ** 2 - x2 + x3 ** 2,
        20 - x1 - x2 ** 2 - x3 ** 2]


def constr2(x):
        x1, x2, x3 = x
        return [-x1 - x2 ** 2 + 2, x2 + 2 * x3 ** 2 - 3]


con1 = {'type': 'ineq', 'fun': constr1} # 键值对 type: 'ineq' 表示不等式约束
con2 = {'type': 'eq', 'fun': constr2} # 键值对 type: 'eq' 表示表示等式约束
con = [con1, con2]
bd = [(0, np.inf) for i in range(3)] # 定义变量的边界， np.inf表示正无穷大
res = minimize(obj, np.random.randn(3), constraints=con, bounds=bd)
print(res)
```

多目标规划

1.模型形式

$$
\begin{aligned}
max\ f(x) = [f_1(x), f_2(x), \ldots, f_m(x)]^T, \\
s.t. \begin{cases} g_i(x) \le 0, & i = 1, 2, \ldots, p, \\ h_j(x) = 0, & j = 1, 2, \ldots, q. \end{cases}
\end{aligned}
$$

多目标规划的可行域（决策空间）：

$$\Omega = \{x\ |g_i(x) \le 0, i = 1, \ldots, p; h_i(x) = 0, j = 1, \ldots, q\}$$

多目标规划的像集（目标空间）：

$$f(\Omega) = \{f(x)\ |x \in \Omega\}$$

若$\bar{x} \in \Omega$，且

$$f_i(\bar{x}) \le f_i(x), \quad i = 1, 2, \ldots, m.$$

恒成立，则称 $\bar{x}$ 为多目标规划问题的绝对最优解.

若$\bar{x} \in \Omega$，且不存在$x \in \Omega$，使得

$$f_i(x) \le f_i(\bar{x}), \quad i = 1, 2, \ldots, m.$$

且至少有一个

$$f_i(x) < f_j(\bar{x}),$$

则称 $\bar{x}$ 为多目标问题的Pareto有效解.

2.解法

（1）线性加权法

确定权值

$$0 \le w_i \le 1, i = 1, 2, \ldots, m\ ; \quad \sum_{i=1}^{m} = 1$$

改写目标函数为

$$min\ \sum_{i=1}^{m} w_i f_i(x)$$

转化为一般（非）线性规划问题.

（2）ε 约束法

选择一个主要的参考目标 $f_k(x)$ ，将其他 $m-1$个目标函数放入约束条件中，
即在约束条件中添加

$$f_i(x) \le \varepsilon_i, \quad i = 1, 2, \ldots, k-1, k+1, \ldots, m.$$

其中 ε_i 为常数.

（3）理想点法

以单个目标最优质为该目标的理想值，使每个目标函数值与理想值的差的加权平方和最小.

首先求出每个目标函数的理想值，即

$$f_i^* = \underset{x \in \Omega}{min}\ f_i(x), \quad i = 1, 2, \ldots, m,$$

然后求评价函数，即每个目标与理想值的差的加权平方和的最优值，即

$$\underset{x \in \Omega}{min}\ \sum_{i=1}^{m} w_i(f_i - f_i^*)^2$$

该方法需要求$m+1$个单目标规划.

（4）优先级法

按照优先级高低来求解目标函数的最优值，在确保优先级高的目标函数的值不低于最优值的情况下，再求优先级低的目标函数的最优值.

确定优先级后，求第一级单一目标函数的最优值，即

$$f_i^* = \min_{x \in \Omega} f_1(x),$$

然后以第一级单一目标函数的最优值为约束，求第二级目标函数的最优值，即

$$\min f_2(x),$$
$$s.t. \begin{cases} f_i(x) = f_1^*, \\ x \in \Omega. \end{cases}$$

以此类推.

例如:
求解下列多目标规划问题的有效解:

$$\min \{-2x_1 + 3x_2,\ x_1 + 2x_2\}$$
$$s.t. \begin{cases} 0.5x_1 + 0.25x_2 \le 8, \\ 0.2x_1 + 0.2x_2 \le 4, \\ x_1 + 5x_2 \le 72, \\ x_1 + x_2 \ge 10, \\ x_1, x_2 \ge 0. \end{cases}$$

解法:

```
import numpy as np
import cvxpy as cp

x = cp.Variable(2, pos=True)
# 线性加权法，取两者权重相同，均为0.5
c1 = np.array([-2, -3])
c2 = np.array([1, 2])
a = np.array([[0.5, 0.25], [0.2, 0.2], [1, 5], [-1, -1]])
b = np.array([8, 4, 72, -10])
obj = cp.Minimize(0.5 * (c1 + c2) @ x)
con = [a @ x <= b]
prob = cp.Problem(obj, con)
prob.solve(solver='GLPK_MI')
print('最优解为', x.value)
print('最优值为', prob.value)
# 理想点法
obj1 = cp.Minimize(c1 @ x)
prob1 = cp.Problem(obj1, con)
prob1.solve(solver='GLPK_MI')
v1 = prob1.value # 第一个目标函数的最优值
obj2 = cp.Minimize(c2 @ x)
prob2 = cp.Problem(obj2, con)
```

```python
prob2.solve(solver='GLPK_MI')
v2 = prob2.value # 第二个目标函数的最优值
obj3 = cp.Minimize((c1 @ x - v1) ** 2 + (c2 @ x - v2) ** 2)
prob3 = cp.Problem(obj3, con)
prob3.solve(solver='CVXOPT')
print('最优值为', v1, v2)
print('最优解为', x.value)
# 优先级法
con.append(c1 @ x == v1)
prob4 = cp.Problem(obj2, con)
prob4.solve(solver='GLPK_MI')
x3 = x.value
print('最优解为', x3)
```