第十二章 优化算法

第 35 讲 约束优化算法

黄定江

DaSE @ ECNU

djhuang@dase.ecnu.edu.cn

1. 35.1 可行方向法
2. 35.2 制约函数法：外点法
3. 35.3 制约函数法：内点法

1. 35.1 可行方向法
2. 35.2 制约函数法：外点法
3. 35.3 制约函数法：内点法

35.1.1 约束优化

实际中遇到的大多数优化问题都是约束优化，它的一般形式为

$$\begin{aligned} &\min\ f_0(\boldsymbol{x}) \\ &\text{s.t. } f_i(\boldsymbol{x}) \le 0, \quad i = 1, 2, \cdots, m \\ &\quad\ h_j(\boldsymbol{x}) = 0, \quad j = 1, 2, \cdots, p \end{aligned} \tag{1}$$

另外，一些特定方法可能会针对于求解仅含有不等式约束优化问题：

$$\begin{aligned} &\min\ f_0(\boldsymbol{x}) \\ &\text{s.t. } f_i(\boldsymbol{x}) \le 0, \quad i = 1, 2, \cdots, m \end{aligned} \tag{2}$$

或者是仅含有等式约束优化问题：

$$\begin{aligned} &\min\ f_0(\boldsymbol{x}) \\ &\text{s.t. } h_j(\boldsymbol{x}) = 0, \quad j = 1, 2, \cdots, p \end{aligned} \tag{3}$$

- 求解约束优化问题要比无约束优化问题困难。
- 不仅要使目标函数值在每次迭代有所下降，而且还要时刻注意解的可行性问题。
- 通常可采用以下方法：将无约束优化问题的求解算法自然推广到约束优化问题；将约束问题化为逐序列的无约束问题；以及将复杂问题变换为较简单问题的其它方法。
- 本讲主要介绍：可行方向法、外点法（二次罚函数法、增广拉格朗日法）和内点法（倒数障碍函数法、对数障碍函数法）。

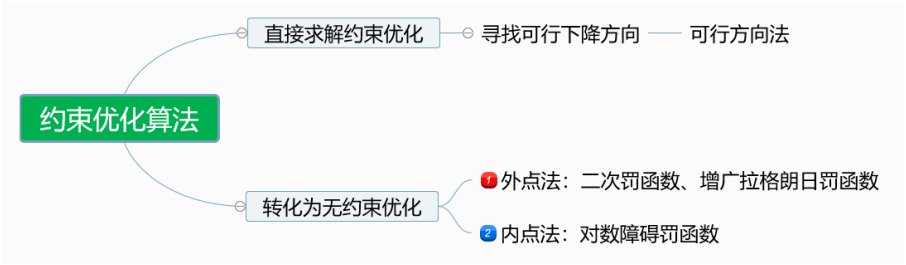

约束优化算法
直接求解约束优化
寻找可行下降方向
可行方向法
转化为无约束优化
1 外点法：二次罚函数、增广拉格朗日罚函数
2 内点法：对数障碍罚函数

图 1: 约束优化算法概览

35.1.2 下降方向

现在介绍求解不等式约束优化问题(2)的可行方向法。下面对约束优化问题相关的概念进行界定：

定义 1

考虑约束优化的某一可行点 $\boldsymbol{x}^{(0)}$，对该点的任一方向 $\boldsymbol{d}$ 来说，若存在实数 $\lambda_0^{'}>0$，使对任意 $\lambda \in [0,\lambda_0^{'}]$ 均有

$$f_0(\boldsymbol{x}^{(0)}+\lambda \boldsymbol{d}) < f_0(\boldsymbol{x}^{(0)})$$

就称方向 $\boldsymbol{d}$ 为 $\boldsymbol{x}^{(0)}$ 点的一个**下降方向**。

35.1.3 可行下降方向

由于约束优化涉及到一个关键的问题是解的可行性问题，所以还有如下概念：

定义 2

假定 $\boldsymbol{x}^{(0)}$ 是待求解的约束优化问题的一个可行点，现考虑此点的某一方向 $\boldsymbol{d}$，若存在实数 $\lambda_0 > 0$，使对于任意 $\lambda \in [0, \lambda_0]$ 均有

$$\boldsymbol{x}^{(0)} + \lambda \boldsymbol{d} \tag{4}$$

满足约束条件，则称方向 $\boldsymbol{d}$ 是 $\boldsymbol{x}^{(0)}$ 点的一个**可行方向**。

如果方向 $\boldsymbol{d}$ 既是 $\boldsymbol{x}^{(0)}$ 点的可行方向，又是这个点的下降方向，就称它是该点的**可行下降方向**。

35.1.4 有效约束

对于含有不等式约束优化的一个可行解 $\boldsymbol{x}^{(0)}$，在不等式约束条件 $f_i(\boldsymbol{x}) \leq 0$ 下可行解处仅存在两种可能：

- 情形一：$f_i(\boldsymbol{x}^{(0)}) < 0$。这时点 $\boldsymbol{x}^{(0)}$ 不是处于由这一约束条件形成的可行域边界上，因而这一约束对 $\boldsymbol{x}^{(0)}$ 点的微小摄动不起限制作用。从而称这个约束条件是 $\boldsymbol{x}^{(0)}$ 点的**不起作用约束**（或无效约束）；
- 情形二：$f_i(\boldsymbol{x}^{(0)}) = 0$。这时 $\boldsymbol{x}^{(0)}$ 点处于该约束条件形成的可行域边界上，它对 $\boldsymbol{x}^{(0)}$ 的摄动起到了某种限制作用。故称这个约束是 $\boldsymbol{x}^{(0)}$ 点的**起作用约束**（有效约束）。

35.1.5 寻找可行下降方向

若假定 $f_0(\boldsymbol{x})$ 和 $f_i(\boldsymbol{x})$ 具有一阶连续偏导数。有了光滑性的假设，可以对可行下降方向的理解进一步深化，以便设计有效的求解方法。

在无约束优化中，我们知道将目标函数 $f_0(\boldsymbol{x})$ 在点 $\boldsymbol{x}^{(0)}$ 处作一阶泰勒展开，可得满足条件

$$\nabla f_0(\boldsymbol{x}^{(0)})^T \boldsymbol{d} < 0 \tag{5}$$

的方向 $\boldsymbol{d}$ 必为 $\boldsymbol{x}^{(0)}$ 点的**下降方向**。

在光滑性假设下，可行方向应该满足什么条件？

若 $\boldsymbol{d}$ 是可行点 $\boldsymbol{x}^{(0)}$ 处的任一**可行方向**（图2），则对该点的所有有效约束均有

$$-\nabla f_i(\boldsymbol{x}^{(0)})^T \boldsymbol{d} \geq 0, \quad i \in J \tag{6}$$

其中 J 为 $\boldsymbol{x}^{(0)}$ 这个点所有起作用约束下标的集合。

注：这是因为它需要保持有效约束函数不会上升，从而保证下一个迭代点仍然可行（≤ 0）。

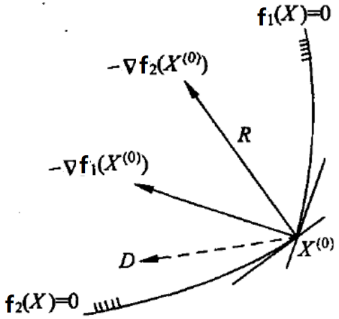


图 2

- 因此，同时满足式(5)和(6)的方向一定是可行下降方向。
- 如果 $\boldsymbol{x}^{(0)}$ 点不是极小点，继续寻优时的搜索方向就应从该点的可行下降方向中去找。
- 显然，若某点存在可行下降方向，它就不会是极小点。另外，若某点为极小点，则在该点不存在可行下降方向，即如下定理所述。

定理 1

设 $\boldsymbol{x}^*$ 是不等式约束优化问题(2)的一个局部极小点，目标函数 $f_0(\boldsymbol{x})$ 在 $\boldsymbol{x}^*$ 处可微，而且

$f_i(\boldsymbol{x})$ 在 $\boldsymbol{x}^*$ 处可微，当 $i \in J$

$f_i(\boldsymbol{x})$ 在 $\boldsymbol{x}^*$ 处连续，当 $i \notin J$

则在 $\boldsymbol{x}^*$ 点不存在可行下降方向，从而不存在向量 $\boldsymbol{d}$ 同时满足：

$$\begin{cases} \nabla f_0(\boldsymbol{x}^*)^T \boldsymbol{d} < 0 \\ -\nabla f_i(\boldsymbol{x}^*)^T \boldsymbol{d} > 0, \quad i \in J \end{cases} \tag{7}$$

证明.

事实上，通过反证法即可证明。若存在满足式(7)的方向 $\boldsymbol{d}$，则通过泰勒展开可以发现，在该点的一个小的邻域内，沿该方向搜索一定可找到更好的可行点。从而，与 $\boldsymbol{x}^*$ 为局部极小点的假设矛盾。 □

注：

- 式(7)就是下降方向(5)和可行方向(6)的交集。
- 式(7)的几何意义：满足该条件的方向 $\boldsymbol{d}$，与点 $\boldsymbol{x}^*$ 处目标函数负梯度方向和有效约束函数负梯度方向的夹角均为锐角。

确定搜索方向

为了设计出求解方法，需要解决第 k 个迭代可行点 $\boldsymbol{x}^{(k)}$ 处搜索方向的问题。显然，此处的搜索方向应当从式(7)可行下降方向中寻找，即从下述不等式组中确定向量 $\boldsymbol{d}$:

$$\begin{cases} \nabla f_0(\boldsymbol{x}^{(k)})^T \boldsymbol{d} < 0 \\ -\nabla f_i(\boldsymbol{x}^{(k)})^T \boldsymbol{d} > 0, \quad i \in J \end{cases} \tag{8}$$

如果在 $\boldsymbol{x}^{(k)}$ 处满足可行下降的方向有多个，这时就需要**思考**如下问题：我们应该选择哪一个方向？无约束优化中的梯度下降法，能否给我们一些启示呢？

在梯度下降法中，我们期望选择的搜索方向使得目标函数值最快地下降，因此得到了搜索方向为负梯度方向。同理，在此同样希望目标函数值尽快地下降，但是这里无法给出一个解析方向。需要转换为如下优化问题：

$$\begin{aligned} \min \quad & \eta \\ \text{s.t.} \quad & \nabla f_0(\boldsymbol{x}^{(k)})^T \boldsymbol{d} \leq \eta \\ & \nabla f_i(\boldsymbol{x}^{(k)})^T \boldsymbol{d} \leq \eta, \quad i \in J(\boldsymbol{x}^{(k)}) \\ & \eta \leq 0 \end{aligned} \tag{9}$$

注：幸运地，这是一个线性规划问题。

上述优化问题还存在一个不足，就是对方向 $\boldsymbol{d}$ 做缩放仍然是满足约束，从而导致无限的最优解。实际上，我们只关注的是这个方向（分量的相对大小）。所以还需加一些限制：

$$
\begin{aligned}
\min \quad & \eta \\
\text{s.t.} \quad & \nabla f_0(\boldsymbol{x}^{(k)})^T \boldsymbol{d} \leq \eta \\
& \nabla f_i(\boldsymbol{x}^{(k)})^T \boldsymbol{d} \leq \eta, \quad i \in J(\boldsymbol{x}^{(k)}) \\
& -1 \leq d_i \leq 1, \quad i = 1, 2, \cdots, n \\
& \eta \leq 0
\end{aligned}
\tag{10}
$$

其中 $d_i (i = 1, 2, \cdots, n)$ 为向量 $\boldsymbol{d}$ 的分量。

- 将线性规划式(10)的最优解记为 $(\boldsymbol{d}^{(k)}, \eta_k)$，如果求出的 $\eta_k = 0$，说明在 $\boldsymbol{x}^{(k)}$ 点不存在可行下降方向，在 $\nabla f_i(\boldsymbol{x}^{(k)})$（此处 $i \in J(\boldsymbol{x}^{(k)})$）线性无关的条件下，$\boldsymbol{x}^{(k)}$ 满足 KKT 条件。
- 若解出的 $\eta_k < 0$，则得到可行下降方向 $\boldsymbol{d}^{(k)}$，这就是我们所要的搜索方向。

35.1.6 可行方向法具体步骤

这样便可以总结出不等式约束优化问题(2)的**可行方向法**的迭代步骤如下：

1. 确定允许误差 $\varepsilon_1 > 0$ 和 $\varepsilon_2 > 0$，选初始近似点 $\boldsymbol{x}^{(0)}$ 满足约束条件，并令 $k := 0$；
2. 确定起作用约束指标集

$$J(\boldsymbol{x}^{(k)}) = \{i | f_i(\boldsymbol{x}^{(k)}) = 0, 1 \leq i \leq m\}$$

(1) 若 $J(\boldsymbol{x}^{(k)}) = \varnothing$（$\varnothing$ 为空集），而且 $\|\nabla f_0(\boldsymbol{x}^{(k)})\| \leq \varepsilon_1$，停止迭代，得点 $\boldsymbol{x}^{(k)}$；

(2) 若 $J(\boldsymbol{x}^{(k)}) = \varnothing$，但 $\|\nabla f_0(\boldsymbol{x}^{(k)})\| > \varepsilon_1$，则取 $\boldsymbol{d}^{(k)} = -\nabla f_0(\boldsymbol{x}^{(k)})$，然后转向第 5 步；

(3) 若 $J(\boldsymbol{x}^{(k)}) \neq \varnothing$，转下一步；

3. 求解线性规划

$$\begin{cases} \min \eta \\ \quad \nabla f_0(\boldsymbol{x}^{(k)})^T \boldsymbol{d} \le \eta \\ \quad \nabla f_i((\boldsymbol{x}^{(k)})^T \boldsymbol{d}) \le \eta, \quad i \in J(\boldsymbol{x}^{(k)}) \\ \quad -1 \le d_i \le 1, \quad i = 1, 2, \cdots, n \\ \quad \eta \le 0 \end{cases}$$

设它的最优解是 $(\boldsymbol{d}^{(k)}, \eta_k)$;

4. 检验是否满足

$$|\eta_k| \le \varepsilon_2$$

若满足则停止迭代，得到点 $\boldsymbol{x}^{(k)}$；否则，以 $\boldsymbol{d}^{(k)}$ 为搜索方向，并转下一步；

5. 解下述一维优化问题

$$\lambda_k: \min_{0\le\lambda\le\overline{\lambda}} f_0(\boldsymbol{x}^{(k)}+\lambda\boldsymbol{d}^{(k)})$$

此处

$$\overline{\lambda}=\max\{\lambda|f_i(\boldsymbol{x}^{(k)}+\lambda\boldsymbol{d}^{(k)})\le 0,\quad i=1,2,\cdots,m\}$$

6. 令

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}+\lambda_k\boldsymbol{d}^{(k)}$$

$$k:=k+1$$

转回第 2 步。

注：由于机器学习中较少使用可行方向法，因此，这里仅介绍了不等式约束情形下的可行方向法。对于更为一般的情形（包括等式约束），限于篇幅，这里不做拓展。实际上，通过简单地思考过程，可以很容易地将该方法推广至线性等式约束的情形。

例 1

用可行方向法解下述约束优化问题

$$\begin{aligned} \max \quad & 4x_1 + 4x_2 - x_1^2 - x_2^2 \\ s.t. \quad & x_1 + 2x_2 \leq 4 \end{aligned}$$

解

先将该约束优化问题写成

$$\begin{aligned} \min \quad & f_0(\boldsymbol{x}) = -(4x_1 + 4x_2 - x_1^2 - x_2^2) \\ s.t. \quad & f_1(\boldsymbol{x}) = x_1 + 2x_2 - 4 \leq 0 \end{aligned}$$

取初始可行点 $\boldsymbol{x}^{(0)} = (0,0)^T, f_0(\boldsymbol{x}^{(0)}) = 0$

$$\nabla f_0(\boldsymbol{x}) = \begin{pmatrix} 2x_1 - 4 \\ 2x_2 - 4 \end{pmatrix}, \quad \nabla f_0(\boldsymbol{x}^{(0)}) = \begin{pmatrix} -4 \\ -4 \end{pmatrix}$$

$$\nabla f_1(\boldsymbol{x}) = (1,2)^T$$

$f_1(\boldsymbol{x}^{(0)}) = -4 < 0$，从而 $J(\boldsymbol{x}^{(0)}) = \varnothing$（空集）。由于

$$\|\nabla f_0(\boldsymbol{x}^{(0)})\|^2 = (-4)^2 + (-4)^2 = 32$$

所以 $\boldsymbol{x}^{(0)}$ 不是（近似）极小点。现取搜索方向

$$\boldsymbol{d}^{(0)} = -\nabla f_0(\boldsymbol{x}^{(0)}) = (4,4)^T$$

从而

$$\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} + \lambda \boldsymbol{d}^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \lambda \begin{pmatrix} 4 \\ 4 \end{pmatrix} = \begin{pmatrix} 4\lambda \\ 4\lambda \end{pmatrix}$$

将其代入约束条件，并令 $f_1(\boldsymbol{x}^{(1)}) = 0$，解得 $\overline{\lambda} = 1/3$。

$$f_0(\boldsymbol{x}^{(1)}) = -16\lambda - 16\lambda + 16\lambda^2 + 16\lambda^2 = 32\lambda^2 - 32\lambda$$

令 $f_0(\boldsymbol{x}^{(1)})$ 对 λ 的导数等于零，解得 $\lambda = 1/2$。因 λ 大于 $\overline{\lambda}(\overline{\lambda} = 1/3)$，故取 $\lambda_0 = \overline{\lambda} = 1/3$。

$$\boldsymbol{x}^{(1)} = \left(\frac{4}{3}, \frac{4}{3}\right)^T, \quad f_0(\boldsymbol{x}^{(1)}) = -\frac{64}{9}$$

$$\nabla f_0(\boldsymbol{x}^{(1)}) = \left(-\frac{4}{3}, -\frac{4}{3}\right)^T, \quad f_1(\boldsymbol{x}^{(1)}) = 0$$

现构成下述线性规划问题

$$\begin{aligned}
&\min\ \eta \\
&\text{s.t.}\ -\frac{4}{3}d_1-\frac{4}{3}d_2\le\eta \\
&\quad d_1+2d_2\le\eta \\
&\quad -1\le d_1\le 1,\quad -1\le d_2\le 1 \\
&\quad \eta\le 0
\end{aligned}$$

从而得到，$\eta=-4/10$，搜索方向

$$\boldsymbol{d}^{(1)}=\begin{pmatrix}d_1\\d_2\end{pmatrix}=\begin{pmatrix}1.0\\-0.7\end{pmatrix}$$

由此

$$\boldsymbol{x}^{(2)}=\boldsymbol{x}^{(1)}+\lambda\boldsymbol{d}^{(1)}=\begin{pmatrix}4/3+\lambda\\4/3-0.7\lambda\end{pmatrix}$$

$$f_0(\boldsymbol{x}^{(2)})=1.49\lambda^2-0.4\lambda-7.111$$

令 $\frac{\mathrm{d}f(\boldsymbol{x}^{(2)})}{\mathrm{d}\lambda}=0$，得到 $\lambda=0.134$。现暂用该步长，算出

$$\boldsymbol{x}^{(2)}=\begin{pmatrix}4/3+0.134\\4/3-0.7\times 0.134\end{pmatrix}=\begin{pmatrix}1.467\\1.239\end{pmatrix}$$

因 $f_1(\boldsymbol{x}^{(2)})=-0.055<0$，上面算出的 $\boldsymbol{x}^{(2)}$ 为可行点，说明选取 $\lambda_1=0.134$ 正确。继续迭代下去，可得最优解为

$$\boldsymbol{x}^*=(1.6,1.2)^T,\quad f_0(\boldsymbol{x}^*)=-7.2.$$

原问题的最优解不变，其原问题目标函数值为 $-f_0(\boldsymbol{x}^*)=7.2$。

1 35.1 可行方向法

2 35.2 制约函数法：外点法

3 35.3 制约函数法：内点法

本节介绍求解约束优化问题的**制约函数法**。这种方法可将约束优化问题的求解，转化为求解一系列无约束优化问题。因而也称这种方法为序列无约束极小化技术，简记为 SUMT (sequential unconstrained minimization technique)。

- 常用的制约函数可分为两类：**罚函数**（penalty function）和**障碍函数**（barrier function）。
- 对应于这两种函数的 SUMT 可分为：**外点法**和**内点法**。

35.2.1 外点法

约束优化问题相比于无约束优化问题的难点，在于不能简单地将负梯度方向作为搜索方向。直观地想法，

- 通过将约束项转化为一种惩罚项。违背约束时，进行惩罚；反之，不惩罚。
- 将惩罚项添加到目标函数中，从而将约束优化问题转化为无约束优化问题进行求解。
- 当惩罚的力度足够大时，约束自然需要得到满足，不然无法极小化目标函数。

下面先考虑等式约束的转化，构造一个函数 $\phi(t)$

$$\phi(t)=\begin{cases}0, & \text{当} t=0\\ \infty, & \text{当} t\neq 0\end{cases} \tag{11}$$

现把 $h_j(\boldsymbol{x})$ 视为 t，显然

- 当 $\boldsymbol{x}$ 满足约束条件时，　　$\phi(h_j(\boldsymbol{x}))=0,\quad j=1,2,\cdots,p;$
- 当 $\boldsymbol{x}$ 不满足约束条件时，　　$\phi(h_j(\boldsymbol{x}))=\infty$。

现在考虑不等式约束的转化，构造一个函数 $\psi(t)$

$$\psi(t)=\begin{cases}0, & \text{当} t\geq 0\\ \infty, & \text{当} t<0\end{cases} \tag{12}$$

现把 $-f_i(\boldsymbol{x})$ 视为 t，显然

- 当 $\boldsymbol{x}$ 满足约束条件时，　$\psi(-f_i(\boldsymbol{x}))=0, \quad i=1,2,\cdots,m;$
- 当 $\boldsymbol{x}$ 不满足约束条件时，　$\psi(-f_i(\boldsymbol{x}))=\infty$。

再构造罚函数

$$P(\boldsymbol{x}) = f_0(\boldsymbol{x}) + \sum_{j=1}^{p} \phi(h_j(\boldsymbol{x})) + \sum_{i=1}^{m} \psi(-f_i(\boldsymbol{x})) \tag{13}$$

就转换成求解无约束问题

$$\min \ P(\boldsymbol{x}) \tag{14}$$

- 若该问题有解，假定其解为 $\boldsymbol{x}^*$，则由式(11)和式(12)知应有 $\phi(h_j(\boldsymbol{x}^*)) = 0$ 和 $\psi(-f_i(\boldsymbol{x}^*)) = 0$。
- 这就是说点 $\boldsymbol{x}^*$ 满足约束条件。因而，$\boldsymbol{x}^*$ 不仅是问题式(14)的极小解，它也是原约束优化问题(1)的极小解。
- 这样一来，确实就把有约束优化问题(1)的求解化成了求解无约束问题(14)。

35.2.2 二次罚函数法

显然，用上述方法构造的函数存在不足，$\phi(t)$ 和 $\psi(t)$ 在 $t=0$ 处不连续，更没有导数。为此，将它们修改为

$$\phi(t)=\begin{cases}0, & \text{当} t=0\\ t^2, & \text{当} t\neq 0\end{cases} \quad \text{以及} \quad \psi(t)=\begin{cases}0, & \text{当} t\geq 0\\ t^2, & \text{当} t<0\end{cases} \tag{15}$$

修改后的函数 $\phi(t)$, $\psi(t)$，当 $t=0$ 时导数等于零，而且 $\phi(t)$, $\psi(t)$ 和 $\phi^{'}(t)$, $\psi^{'}(t)$ 对任意 t 都连续。

注意：

- 当 $\boldsymbol{x}$ 满足约束条件时仍有

$$\sum_{j=1}^{p} \phi(h_j(\boldsymbol{x})) = 0, \quad \sum_{i=1}^{m} \psi(-f_i(\boldsymbol{x})) = 0$$

- 但是，当 $\boldsymbol{x}$ 不满足约束条件时

$$0 < \sum_{j=1}^{p} \phi(h_j(\boldsymbol{x})) < \infty$$

$$0 < \sum_{i=1}^{m} \psi(-f_i(\boldsymbol{x})) < \infty$$

因此，可能存在因为惩罚的力度不够，导致无约束优化问题的解不满足原约束条件的情形发生。故我们需要对无约束优化问题(14)的目标函数 $P(\boldsymbol{x})$ 进行适当的改变。

定义 3

对一般的约束优化问题(1)，定义如下**二次罚函数**：

$$P(\boldsymbol{x}, M) = f_0(\boldsymbol{x}) + M\left[\sum_{j=1}^{p}[h_j(\boldsymbol{x})]^2 + \sum_{i=1}^{m}[\min(0, -f_i(\boldsymbol{x}))]^2\right] \tag{16}$$

其中等式第二项为惩罚项，$M > 0$ 为罚因子。

- 若求得的无约束优化问题的最优解 $\boldsymbol{x}(M)$ 满足约束条件，则它必定是原问题的极小解。事实上，对于所有满足约束条件的 $\boldsymbol{x}$

$$f_0(\boldsymbol{x}) + M\left[\sum_{j=1}^{p}\phi(h_j(\boldsymbol{x})) + \sum_{i=1}^{m}\psi(-f_i(\boldsymbol{x}))\right] = P(\boldsymbol{x}, M)$$

$$\geq P(\boldsymbol{x}(M), M) = f_0(\boldsymbol{x}(M))$$

  即当 $\boldsymbol{x}$ 满足约束条件时，有 $f_0(\boldsymbol{x}) \geq f_0(\boldsymbol{x}(M))$。

- 虽然有了罚因子 M，但仍然可能出现最优解不满足约束条件。所以需要不断地增大罚因子 M 的值，迫使最优解满足所有约束条件，下面给出了图示。

图3示出了一个二次罚函数的例子：

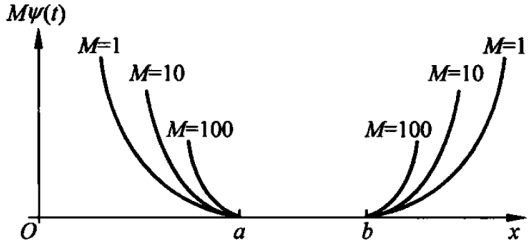


图 3: 左半部表示约束条件 $f_1(x) = a - x \leq 0$，右半部则表示 $f_2(x) = x - b \leq 0$

若对于某一个**罚因子**M，例如 M_1，$\boldsymbol{x}(M_1)$ 不满足约束条件，就加大罚因子的值。随着 M 值得增加，惩罚函数中的惩罚项所起的作用随之增大。$\min\ P(\boldsymbol{x}, M)$ 的解 $\boldsymbol{x}(M)$ 与约束集的“距离”就越来越近，当

$$0 < M_1 < M_2 < \cdots < M_k < \cdots$$

趋于无穷大时，点列 $\{\boldsymbol{x}(M_k)\}$ 就从可行域的外部趋于原问题式(1)的极小点 $\boldsymbol{x}_{\min}$。

借此可对外点法作如下经济解释：

- 把目标函数 $f(\boldsymbol{x})$ 看成“价格”，约束条件看成某种“规定”，采购人可在规定范围内购置最便宜的东西。
- 此外对违反规定制定了一种“**罚款**”政策，若符合规定，罚款为零；否则，要收罚款。
- 采购人付出的总代价应是价格和罚款的总和。采购者的目标是使总代价最小，这就是上述的无约束问题。
- 当罚款规定不够苛刻时，采购人可能存在一些**投机行为**，可能违反部分规定，使得总代价最小。
- 当罚款规定得很苛刻时，违反规定支付的罚款很高，这就**迫使采购人符合规定**。在数学上表现为罚因子 M_k 足够大时，上述无约束问题的最优解应满足约束条件，而成为约束条件的最优解。

二次罚函数法具体步骤

最后，将**二次罚函数法**的迭代步骤总结如下：

Algorithm 1 二次罚函数法

1: 给定初值 $\boldsymbol{x}^{(0)}$。取 $M_1 > 0$（例如说取 $M_1 = 1$），允许误差 $\varepsilon > 0$，并令 $k := 0$；

2: 以 $\boldsymbol{x}^{(k)}$ 为初始点，求无约束优化问题的最优解：$\boldsymbol{x}^{(k+1)} = \arg\min_{\boldsymbol{x}} P(\boldsymbol{x}, M_k)$, 式中

$$P(\boldsymbol{x}, M_k) = f_0(\boldsymbol{x}) + M_k \left[\sum_{j=1}^{p} [h_j(\boldsymbol{x})]^2 + \sum_{i=1}^{m} [\min(0, -f_i(\boldsymbol{x}))]^2 \right]$$

3: 若对某一个 $j(1 \le j \le p)$ 或 $i(1 \le i \le m)$ 有

$$|h_j(\boldsymbol{x}^{(k)})| \ge \varepsilon,\ f_i(\boldsymbol{x}^{(k)}) \ge \varepsilon$$

则取 $M_{k+1} > M_k$（例如，$M_{k+1} = cM_k$，（$c > 1$）），令 $k := k + 1$，并转向第 2 步。否则，停止迭代，得

$$\boldsymbol{x}_{\min} \approx \boldsymbol{x}^{(k)}$$

例 2

求解约束优化问题

$$\begin{aligned} \min\ & f_0(\boldsymbol{x}) = x_1 + x_2 \\ s.t.\ & f_1(\boldsymbol{x}) = x_1^2 - x_2 \le 0 \\ & f_2(\boldsymbol{x}) = -x_1 \le 0 \end{aligned}$$

解

构造罚函数

$$P(\boldsymbol{x}, M) = x_1 + x_2 + M\{[\min(0, (-x_1^2 + x_2))]^2 + [\min(0, x_1)]^2\}$$

$$\frac{\partial P}{\partial x_1} = 1 + 2M[\min(0, (-x_1^2 + x_2)(-2x_1))] + 2M[\min(0, x_1)]$$

$$\frac{\partial P}{\partial x_2} = 1 + 2M[\min(0, (-x_1^2 + x_2))]$$

对于不满足约束条件的点 $\boldsymbol{x} = (x_1, x_2)^T$，有

$$-x_1^2 + x_2 < 0, \quad x_1 < 0$$

令

$$\frac{\partial P}{\partial x_1} = \frac{\partial P}{\partial x_2} = 0$$

得 $\min\ P(\boldsymbol{x}, M)$ 的解为

$$\boldsymbol{x}(M) = \left(-\frac{1}{2(1+M)}, \left(\frac{1}{4(1+M)^2} - \frac{1}{2M} \right) \right)^T$$

取 $M = 1, 2, 3, 4$，可得出以下结果：

$$M = 1: \quad \boldsymbol{x} = (-1/4, -7/16)^T$$

$$M = 2: \quad \boldsymbol{x} = (-1/6, -2/9)^T$$

$$M = 3: \quad \boldsymbol{x} = (-1/8, -29/192)^T$$

$$M = 4: \quad \boldsymbol{x} = (-1/10, -23/200)^T$$

可知 $\boldsymbol{x}(M)$ 从约束条件外面逐步逼近约束条件的边界，当 $M \to \infty$ 时，$\boldsymbol{x}(M)$ 趋于原问题的极小解 $\boldsymbol{x}_{\min} = (0, 0)^T$（见图4）。

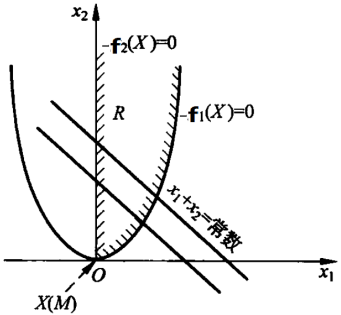


图 4

二次罚函数法存在的不足：

- 易知，为了保证最优解可行，罚因子必须趋于无穷大。
- 实际上，罚因子趋于无穷大时，子问题变得病态而难以求解。

是否能对二次罚函数进行某种修正，使得对于有限的罚因子得到的最优解也是可行的？这就是接下来要介绍的增广拉格朗日函数法。

35.2.3 增广拉格朗日函数法

为方便理解，这里仅介绍等式约束优化问题的增广拉格朗日函数法。该方法是对二次罚函数的一个修正。它是在拉格朗日函数的基础之上，增加约束条件的二次罚函数。定义如下：

定义 4

对等式约束优化问题(3)，定义如下**增广拉格朗日函数**：

$$L_M(\boldsymbol{x}, \boldsymbol{\lambda}) = f_0(\boldsymbol{x}) + \sum_{j=1}^{p} [\lambda_j h_j(\boldsymbol{x})] + M \sum_{j=1}^{p} [h_j(\boldsymbol{x})]^2 \tag{17}$$

其中 $\boldsymbol{\lambda}$ 为拉格朗日乘子，$M > 0$ 仍为罚因子。

因此，希望每次迭代时求解如下无约束优化问题去近似原优化问题：

$$\min_{\boldsymbol{x}} L_{M_k}(\boldsymbol{x}, \boldsymbol{\lambda}^{(k)}).$$

在得到该优化问题的具体迭代步骤前，我们需要思考如下问题：

- 增广拉格朗日函数优化问题的最优解能趋于原等式约束优化问题的最优解？
- 罚因子在每次迭代采用逐步放大时，拉格朗日乘子 $\boldsymbol{\lambda}$ 在每次迭代时需要如何变化？

下面通过对比原问题与增广拉格朗日函数优化问题的最优性条件，来寻找答案。

原等式约束优化问题的最优解 $\boldsymbol{x}^*$ 与相应的拉格朗日乘子 $\boldsymbol{\lambda}^*$，应满足

$$\nabla f_0(\boldsymbol{x}^*) + \sum_{j=1}^{p} [\lambda_j^* \nabla h_j(\boldsymbol{x}^*)] = 0 \tag{18}$$

而增广拉格朗日函数在第 k 次迭代时的最优解 $\boldsymbol{x}^{(k+1)}$，应满足

$$\nabla f_0(\boldsymbol{x}^{(k+1)}) + \sum_{j=1}^{p} [\lambda_j^{(k)} + 2M_k h_j(\boldsymbol{x}^{(k+1)})] \nabla h_j(\boldsymbol{x}^{(k+1)}) = 0 \tag{19}$$

通过对比发现，为使得迭代点列 $\boldsymbol{x}^{(k)}$ 收敛到 $\boldsymbol{x}^*$，则需要保证上述式(18)与式(19)的一致性。因此

$$\lambda_j^* \approx \lambda_j^{(k)} + 2M_k h_j(\boldsymbol{x}^{(k+1)}), \quad j = 1, \cdots, p. \tag{20}$$

从而得到乘子的下一步迭代格式

$$\lambda_j^{(k+1)} = \lambda_j^{(k)} + 2M_k h_j(\boldsymbol{x}^{(k+1)}), \quad j = 1, \cdots, p. \tag{21}$$

直观分析：

- 若 $\boldsymbol{x}^{(k)}$, $\boldsymbol{\lambda}^{(k)}$ 分别收敛到 $\boldsymbol{x}^*$, $\boldsymbol{\lambda}^*$，则由式(20)知

$$h_j(\boldsymbol{x}^{(k+1)}) \approx \frac{1}{2M_k}(\lambda_j^* - \lambda_j^{(k)}).$$

  因此 $h_j(\boldsymbol{x}^{(k)})$ 也将趋于 0。

- 当 λ_j^* 足够接近 $\lambda_j^{(k)}$ 时，允许罚因子不需要足够大（无需趋于无穷大），也能保证约束条件满足。

事实上，我们可以得到如下收敛定理：

定理 2

假设 $\boldsymbol{x}^*$, $\boldsymbol{\lambda}^*$ 分别是原等式约束优化问题(3)的严格局部最小解和相应的拉格朗日乘子，那么，存在足够大的常数 $\overline{M} > 0$ 和足够小的常数 $\delta > 0$，如果对某个 k，有

$$\frac{1}{M_k}\|\boldsymbol{\lambda}^{(k)} - \boldsymbol{\lambda}^*\| < \delta, \quad M_k \geq \overline{M},$$

则

$$\boldsymbol{\lambda}^{(k)} \to \boldsymbol{\lambda}^*, \quad \boldsymbol{x}^{(k)} \to \boldsymbol{x}^*.$$

注：该定理表明不需要 M_k 趋于正无穷。

增广拉格朗日函数法具体步骤

最后，将**增广拉格朗日函数法**的迭代步骤总结如下：

Algorithm 2 增广拉格朗日函数法

1: 给定初值 $\boldsymbol{x}^{(0)}$ 和乘子 $\boldsymbol{\lambda}^{(0)}$。取 $M_0 > 0$，允许精度要求 $\eta_k > 0$ 和约束条件违反误差 $\varepsilon > 0$，并令 $k := 0$；

2: 以 $\boldsymbol{x}^{(k)}$ 为初始点，求无约束优化问题的最优解：

$$\boldsymbol{x}^{(k+1)} = \arg\min_{\boldsymbol{x}} \ L_{M_k}(\boldsymbol{x}, \boldsymbol{\lambda}^{(k)})$$

使得精度满足 $\|\nabla_x L_{M_k}(\boldsymbol{x}^{(k+1)}, \boldsymbol{\lambda}^{(k)})\| \leq \eta_k$.

3: 若对所有 $j(1 \leq j \leq p)$ 有 $|h_j(\boldsymbol{x}^{(k)})| \leq \varepsilon$，则停止迭代，返回近似解 $\boldsymbol{x}^{(k+1)}$, $\boldsymbol{\lambda}^{(k)}$；否则转到下一步。

4: 更新：$\lambda_j^{(k+1)} = \lambda_j^{(k)} + 2M_k h_j(\boldsymbol{x}^{(k+1)})$，$M_{k+1} = cM_k$（$c > 1$），令 $k := k + 1$，并转向第 2 步。

外点法的优劣：

- 函数 $P(\boldsymbol{x}, M)$ 是在 $\mathbb{R}^n$ 上进行优化，初始点可任意选择，这给计算带来了很大方便。而且外点法也可用于非凸函数的最优化。
- 外点法同时适用于含有等式和不等式约束条件的优化问题。
- 如果可行域函数的性质比较复杂，甚至没有定义，这时就无法使用外点法。

应用：二次罚函数法求解低秩矩阵恢复问题

在前面的章节中，我们介绍了低秩矩阵恢复问题（又称矩阵补全问题), 并引入了该问题的形式如下:

$$\begin{aligned} \min \quad & \|\boldsymbol{X}\|_* \\ \text{s.t.} \quad & \boldsymbol{X}_{ij} = \boldsymbol{M}_{ij}, \quad (i,j) \in \Omega, \end{aligned}$$

我们对其中的等式约束引入二次罚函数可以得到

$$\min \quad \|\boldsymbol{X}\|_* + \frac{\sigma}{2} \sum_{(i,j)\in\Omega} \left(\boldsymbol{X}_{ij} - \boldsymbol{M}_{ij}\right)^2.$$

当罚因子 $\sigma = \frac{1}{\mu}$ 时, 上述优化问题转化为如下优化问题

$$\min \quad \mu\|\boldsymbol{X}\|_* + \frac{1}{2} \sum_{(i,j)\in\Omega} \left(\boldsymbol{X}_{ij} - \boldsymbol{M}_{ij}\right)^2. \tag{22}$$

因此我们可以使用罚函数法的策略求解低秩矩阵恢复问题, 具体见算法如下.

Algorithm 3 低秩矩阵恢复的罚函数法

给定初值 $\boldsymbol{X}^0$, 最终参数 μ, 初始参数 μ_0, 因子 $\gamma \in (0,1), k \leftarrow 0$.
while $\mu_k \geqslant \mu$ **do**
  以 $\boldsymbol{X}^k$ 为初值, $\mu = \mu_k$ 为正则化参数求解问题 (22), 得 $\boldsymbol{X}^{k+1}$.
  if $\mu_k = \mu$ **then**
    停止迭代, 输出 $\boldsymbol{X}^{k+1}$.
  else
    更新罚因子 $\mu_{k+1} = \max\{\mu, \gamma\mu_k\}$.
    $k \leftarrow k+1$.
  end if
  $k \leftarrow k+1$
end while

应用：增广拉格朗日函数法求解半定规化问题

考虑半定规划问题：

$$\begin{aligned} \min \quad & \mathsf{Tr}(\boldsymbol{C}\boldsymbol{X}) \\ \text{s.t.} \quad & \mathsf{Tr}(\boldsymbol{A}_j\boldsymbol{X}) = \boldsymbol{b}_j, j = 1, \cdots, p \\ & \boldsymbol{X} \succeq 0 \end{aligned} \tag{23}$$

其中 $\boldsymbol{C}, \boldsymbol{A}_1, \cdots, \boldsymbol{A}_p \in S^n$，$\mathsf{Tr}(\cdot)$ 是迹函数。其对偶问题为:

$$\begin{aligned} & \min_{\boldsymbol{y}\in\mathbb{R}^p} -\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y}, \\ & \text{s.t.} \quad \sum_{j=1}^{p} y_j\boldsymbol{A}_j \preceq \boldsymbol{C}. \end{aligned} \tag{24}$$

我们可以利用增广拉格朗日函数法求解. 对于原始问题, 引入乘子 $\boldsymbol{\lambda} \in \mathbb{R}^p$, 罚因子 σ, 并记 $\mathcal{A}(\boldsymbol{X}) = (\mathsf{Tr}(\boldsymbol{A}_1\boldsymbol{X}), \mathsf{Tr}(\boldsymbol{A}_2\boldsymbol{X}), \cdots, \mathsf{Tr}(\boldsymbol{A}_p\boldsymbol{X}))^{\mathrm{T}}$, 则增广拉格朗日函数为

$$L_\sigma(\boldsymbol{X}, \boldsymbol{\lambda}) = \langle \boldsymbol{C}, \boldsymbol{X} \rangle - \boldsymbol{\lambda}^{\mathrm{T}}(\mathcal{A}(\boldsymbol{X}) - \boldsymbol{b}) + \frac{\sigma}{2}\|\mathcal{A}(\boldsymbol{X}) - \boldsymbol{b}\|_2^2, \quad \boldsymbol{X} \succeq 0.$$

那么, 增广拉格朗日函数法为

$$\begin{cases} \boldsymbol{X}^{k+1} \approx \underset{\boldsymbol{X} \in \mathcal{S}_+^n}{\arg\min} L_{\sigma_k}\left(\boldsymbol{X}, \boldsymbol{\lambda}^k\right), \\ \boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k - \sigma_k\left(\mathcal{A}\left(\boldsymbol{X}^{k+1}\right) - \boldsymbol{b}\right), \\ \sigma_{k+1} = \min\left\{\rho\sigma_k, \bar{\sigma}\right\}. \end{cases}$$

这里, 当迭代收敛时, $\boldsymbol{X}^k$ 和 $\boldsymbol{\lambda}^k$ 分别收敛到问题 (23) 和 (24) 的解。

1 35.1 可行方向法

2 35.2 制约函数法：外点法

3 35.3 制约函数法：内点法

35.3.1 内点法

前面已经提及：如果 $f_0(\boldsymbol{x})$ 在可行域外的性质比较复杂，甚至没有定义，这时就无法使用外点法。现在**思考**：

- 能否设计一种类似于外点法的算法：通过逐序列的无约束优化问题的解，不断地逼近原始约束优化问题的解。
- 同时要求这种方法满足：每次迭代过程始终在可行域内部进行。

注：把取在可行域内部（即既不在可行域外，也不在可行域边界上）的可行点称为**内点**或**严格内点**。

一种直观的想法：仍然使用约束条件改造原目标函数，使得它在原可行域的边界上设置一道“障碍”，使迭代点靠近可行域的边界时，给出的新目标函数值迅速增大，从而使迭代点始终留在可行域内部。这种方法实际上就是接下来要介绍的内点法。

- 改造后的目标函数通常被称为**障碍函数**。
- 满足这种要求的障碍函数，其极小解自然不会在可行域的边界上达到。这是因为虽然可行域是一个闭集，但因极小点不在闭集的边界上，只能在可行域内部取得，因而实际上是具有**无约束性质**的优化问题，可借助于无约束优化的方法进行计算。
- 由于要保持在可行域内部，因此，内点法主要用于不等式约束问题(2)。

根据上述分析，需要将约束优化式(2)转化为下述一系列无约束性质的极小化问题：

$$\min_{\boldsymbol{x}\in R_0} \overline{P}(\boldsymbol{x}, r_k) \tag{25}$$

其中 $\overline{P}(\boldsymbol{x}, r_k)$ 是障碍函数，$R_0 = \{\boldsymbol{x}|f_i(\boldsymbol{x}) < 0, \quad i = 1, 2, \cdots, m\}$。

只要能在可行域边界上建立起“障碍”，我们便可以设计出不同的障碍函数。根据障碍函数的不同，便可得到不同的障碍函数法。

35.3.2 倒数障碍函数

定义 5

对一般的约束优化问题(2)，定义如下**倒数障碍函数**：

$$\overline{P}(\boldsymbol{x}, r_k) = f_0(\boldsymbol{x}) + r_k \sum_{i=1}^{m} \frac{1}{-f_i(\boldsymbol{x})}, \quad (r_k > 0) \tag{26}$$

其中等式第二项为障碍项，$r_k > 0$ 为罚因子。

注：在接近可行域的边界上（即至少有一个 $-f_i(\boldsymbol{x}) \to 0^+$），$\frac{1}{-f_i(\boldsymbol{x})}$ 趋于正无穷大，则 $\overline{P}(\boldsymbol{x}, r_k)$ 趋于正无穷大。

35.3.3 对数障碍函数

定义 6

对一般的约束优化问题(2)，定义如下**对数障碍函数**：

$$\overline{P}(\boldsymbol{x}, r_k) = f_0(\boldsymbol{x}) - r_k \sum_{i=1}^{m} \log(-f_i(\boldsymbol{x})), \quad (r_k > 0) \tag{27}$$

其中等式第二项为障碍项，$r_k > 0$ 为罚因子。

注：在接近可行域的边界上（即至少有一个 $-f_i(\boldsymbol{x}) \to 0^+$），$\log(-f_i(\boldsymbol{x}))$ 趋于负无穷大，则 $\overline{P}(\boldsymbol{x}, r_k)$ 趋于正无穷大。

如果从可行域内部的某一点 $\boldsymbol{x}^{(0)}$ 出发，按无约束极小化方法对式(25)进行迭代（在进行一维搜索时要使用控制步长，以免迭代点跑到 R_0 之外），则随着**障碍因子**r_k 的逐步减小，即

$$r_1 > r_2 > \cdots > r_k > \cdots > 0$$

障碍项所起的作用也越来越小，因而，求出的 $\min\ \overline{P}(\boldsymbol{x}, r_k)$ 的解 $\boldsymbol{x}(r_k)$ 也逐步逼近原问题式(2)的极小解 $\boldsymbol{x}_{\min}$。

35.3.4 内点法具体步骤

现在，将一般的**内点法**的迭代步骤总结如下：

Algorithm 4 内点法

1: 取 $r_1 > 0$（例如取 $r_1 = 1$），允许误差 $\varepsilon > 0$；
2: 找出一可行点 $\boldsymbol{x}^{(0)} \in R_0$，并令 $k = 0$；
3: 构造障碍函数，障碍项可采用倒数障碍函数（式(26)），也可采用对数障碍函数（例如式(27)）；
4: 以 $\boldsymbol{x}^{(k)} \in R_0$ 为初始点，对障碍函数进行无约束极小化（在 R_0 内）：

$$\boldsymbol{x}^{(k+1)} = \arg\min_{\boldsymbol{x} \in R_0} \overline{P}(\boldsymbol{x}, r_k) \tag{28}$$

式中 $\overline{P}(\boldsymbol{x}, r_k)$ 见式(26)或(27)；

Algorithm 4 内点法（续）

1: 检验是否满足收敛准则

$$r_k \sum_{i=1}^{m} \frac{1}{-f_i(\boldsymbol{x}^{(k)})} \leq \varepsilon \quad \text{（倒数障碍函数）}$$

或

$$\left| r_k \sum_{i=1}^{m} \log(-f_i(\boldsymbol{x}^{(k)})) \right| \leq \varepsilon \quad \text{（对数障碍函数）}$$

如满足上述准则，则以 $\boldsymbol{x}^{(k)}$ 为原问题的近似极小解 $\boldsymbol{x}_{\min}$；否则，取 $r_{k+1} < r_k$（例如取 $r_{k+1} = r_k/10$ 或 $r_k/5$），令 $k := k+1$，转向第 3 步继续进行迭代。

注：值得指出的是，收敛准则也可采用不同的形式，例如：

$$\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\| < \varepsilon \quad \text{或} \quad \|f_0(\boldsymbol{x}^{(k)}) - f_0(\boldsymbol{x}^{(k-1)})\| < \varepsilon.$$

例 3

试用内点法求解

$$\min f_0(\boldsymbol{x}) = \frac{1}{3}(x_1+1)^3 + x_2$$
$$s.t.\ f_1(\boldsymbol{x}) = 1 - x_1 \le 0$$
$$f_2(\boldsymbol{x}) = -x_2 \le 0$$

解

构造倒数障碍函数

$$\overline{P}(\boldsymbol{x}, r) = \frac{1}{3}(x_1 + 1)^3 + x_2 + \frac{r}{x_1 - 1} + \frac{r}{x_2}$$

$$\frac{\partial \overline{P}}{\partial x_1} = (x_1 + 1)^2 - \frac{r}{(x_1 - 1)^2} = 0$$

$$\frac{\partial \overline{P}}{\partial x_2} = 1 - \frac{r}{x_2^2} = 0$$

联立解上述两个方程，得

$$x_1(r) = \sqrt{1 + \sqrt{r}}, \quad x_2(r) = \sqrt{r}$$

如此得最优解：

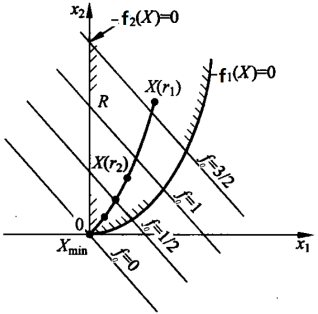
$$\boldsymbol{x}_{\min} = \lim_{r \to 0} \left(\sqrt{1 + \sqrt{r}}, x_2(r) = \sqrt{r} \right)^T = (1, 0)^T$$

由于此例可解析求解，故可如上进行。但很多实际问题不便用解析法，仍需用迭代法求解。

例 4

使用内点法解

$$\min f_0(\boldsymbol{x}) = x_1 + x_2$$
$$s.t.\ f_1(\boldsymbol{x}) = x_1^2 - x_2 \le 0$$
$$f_2(\boldsymbol{x}) = -x_1 \le 0$$



解

构造对数障碍函数如下：

$$\overline{P}(\boldsymbol{x}, r) = x_1 + x_2 - r\log(-x_1^2 + x_2) - r\log x_1$$

各次迭代结果示于表1和图5。

障碍因子	r	$x_1(r)$	$x_2(r)$
r_1	1.000	0.500	1.250
r_2	0.500	0.309	0.595
r_3	0.250	0.183	0.283
r_4	0.100	0.085	0.107
r_5	0.0001	0.000	0.000

Table 1

35.3.5 寻找初始内点

实际上，上述内点法还存在一个**问题**：我们知道，内点法的迭代过程必须由某个内点开始。在处理实际问题时，如果不能找出某个内点作为初始点，内点法的迭代就无法展开。那么，怎样得到初始的内点呢？

注：这在高维空间并不是一件简单的事情。

一个较为直观的想法：

- 先任找一点 $\boldsymbol{x}^{(0)}$ 为初始点，则约束条件只会存在于如下两类：

$$S_0 = \{i | -f_i(\boldsymbol{x}^{(0)}) \leq 0, \quad 1 \leq i \leq m\}$$

$$T_0 = \{i | -f_i(\boldsymbol{x}^{(0)}) > 0, \quad 1 \leq i \leq m\}$$

- 我们期望在寻找的过程中 T_0 内的条件始终满足，而 S_0 中的约束条件趋向满足（即函数值增大）。
- 这恰好等价于，求解将 T_0 作为约束条件，而 S_0 内约束条件的函数之和的相反数作为目标的优化问题。
- 这时将 $\boldsymbol{x}^{(0)}$ 作为初始点，用内点法求解该优化问题得到下一迭代点
- 重复上述步骤，至所有约束条件得到满足，即得到原问题的初始内点。

35.3.6 初始内点具体求解步骤

初始内点的具体迭代步骤如下：

1. 任取一点 $\boldsymbol{x}^{(0)}, r_0 > 0$（例如 $r_0 = 1$），令 $k = 0$；
2. 定出指标集 S_k 及 T_k
$$S_k = \{i | -f_i(\boldsymbol{x}^{(k)}) \leq 0, \quad 1 \leq i \leq m\}, \quad T_k = \{i | -f_i(\boldsymbol{x}^{(k)}) > 0, \quad 1 \leq i \leq m\}$$
3. 检查集合 S_k 是否为空集，若为空集，则 $\boldsymbol{x}^{(k)}$ 在 R_0 内，初始内点找到，迭代停止，否则转向第 4 步；
4. 构造函数
$$\widetilde{P}(\boldsymbol{x}, r_k) = \sum_{i \in S_k} f_i(\boldsymbol{x}) + r_k \sum_{i \in T_k} \frac{1}{-f_i(\boldsymbol{x})}, \quad (r_k > 0)$$
以 $\boldsymbol{x}^{(k)}$ 为初始点，在保持对集合 $\widetilde{R}_k = \{\boldsymbol{x} | -f_i(\boldsymbol{x}) > 0, \quad i \in T_k\}$ 可行的情况下，极小化 $\widetilde{P}(\boldsymbol{x}, r_k)$，即 $\min \widetilde{P}(\boldsymbol{x}, r_k), \quad \boldsymbol{x} \in \widetilde{R}_k,$
得 $\boldsymbol{x}^{(k+1)}, \boldsymbol{x}^{(k+1)} \in \widetilde{R}_k$，转向第 5 步；
5. 令 $0 < r_{k+1} < r_k$（例如说 $r_{k+1} = r_k/10, k := k+1$），转向第 2 步。

35.3.7 应用：对数障碍函数求解网络比率优化问题

网络比率优化问题

- 用 L 个弧或边组成的有向图描述网络，货物或信息包通过边在网络上移动。
- 网络支持 n 个流, 它们的 (非负的) 比率 $x_1, \cdots, x_n$ 是优化变量。每个流沿着网络上一个固定的, 或预先设定的道路（或线路）从源结点向目标结点移动。
- 每条边可以支持多个流通过。每条边上的总交通量等于通过它的所有流量的比率之和。每条边有一个正的容量, 这是在它上面能够通过的总交通量的最大值。
- 可以用下面定义的流 -边关联矩阵 $\boldsymbol{A} \in \mathbf{R}^{L\times n}$ 描述这些边上的容量限制,

$$A_{ij} = \begin{cases} 1 & \text{流} j \text{ 通过边} i \\ 0 & \text{其他情况。} \end{cases}$$

于是边容量约束可以用 $\boldsymbol{Ax} \leq \boldsymbol{c}$ 表示, 其中 c_i 是边 i 上的容量。通常每个道路只通过所有边中的一小部分, 因此 $\boldsymbol{A}$ 是稀疏矩阵。

在网络比率问题中道路是固定的 (并作为问题的参数记录在矩阵 $\boldsymbol{A}$ 中); 变量是流的比率 x_i。目标是选择流的比率使下面的可分效用函数 U 达到最大,

$$U(\boldsymbol{x}) = U_1(x_1) + \cdots + U_n(x_n)$$

我们假定每个 U_i (从而 U) 是凹和非减的。可以把 $U_i(x_i)$ 视为通过以比率 x_i 支持第 i 个流产生的收入; 于是 $U(\boldsymbol{x})$ 是相应的流产生的总收入。**网络比率优化问题**可写成

$$\begin{aligned} \max \quad & U(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{c}, \\ & \boldsymbol{x} \geq 0 \end{aligned}$$

这是一个凸优化问题。

我们用对数障碍方法构造对应的无约束优化问题, 其目标函数为

$$-tU(\boldsymbol{x})-\sum_{i=1}^{L}\log(\boldsymbol{c}-\boldsymbol{A}\boldsymbol{x})_i-\sum_{j=1}^{n}\log x_j.$$

每步迭代我们用 Newton 方法求解对应的无约束优化，确定 Newton 步径 $\Delta\boldsymbol{x}_{\mathrm{nt}}$ 需要求解线性方程组

$$\left(\boldsymbol{D}_0+\boldsymbol{A}^T\boldsymbol{D}_1\boldsymbol{A}+\boldsymbol{D}_2\right)\Delta\boldsymbol{x}_{\mathrm{nt}}=-\boldsymbol{g},$$

其中

$$\begin{aligned}\boldsymbol{D}_0&=-t\,\mathrm{diag}\left(U_1''(\boldsymbol{x}),\cdots,U_n''(\boldsymbol{x})\right)\\ \boldsymbol{D}_1&=\mathrm{diag}\left(1/(\boldsymbol{c}-\boldsymbol{A}\boldsymbol{x})_1^2,\cdots,1/(\boldsymbol{c}-\boldsymbol{A}\boldsymbol{x})_L^2\right)\\ \boldsymbol{D}_2&=\mathrm{diag}\left(1/x_1^2,\cdots,1/x_n^2\right)\end{aligned}$$

是对角矩阵, 而 $\boldsymbol{g}\in\mathbf{R}^n$ 。

我们可以精确描述这个 $n \times n$ 系数矩阵的稀疏结构: 当且仅当流 i 和流 j 共享一条边时才成立

$$\left(\boldsymbol{D}_0 + \boldsymbol{A}^T \boldsymbol{D}_1 \boldsymbol{A} + \boldsymbol{D}_2\right)_{ij} \neq 0.$$

如果道路都比较短, 而每条边只有较少的道路通过, 那么这个矩阵就是稀疏的, 因此其稀疏的 Cholesky 因式分解可以被利用去求解牛顿步径。

注：当 Newton 系统的某些 (不是很多) 行和列稠密时, 我们也可以进行有效的求解。这种情况发生于仅有很少数的流和大量的流相交的时候, 如果比较长的流很少就可能出现这种情况。

本讲小结

可行方向法

- 可行方向、下降方向、有效约束
- 寻找可行下降方向
- 可行方向法

制约函数法

- 外点法：二次罚函数法、增广拉格朗日罚函数法
- 内点法：倒数障碍函数法、对数障碍函数法、寻找初始内点

至此，我们已经介绍完了最优化问题的主要求解方法。本讲主要介绍约束优化的求解方法。实际上，在大数据背景下，传统优化方法面临着新的挑战，我们将在下一讲内容进行探讨。