Dong Wang

现代机器学习技术导论

2018 年 1 月 29 日

Springer

Contents

Chapter 1
机器学习概述

Chapter 2
线性模型

Chapter 3
神经模型

Chapter 4
深度学习

Chapter 5
核方法

Chapter 6
图模型

Chapter 7
非监督学习

Chapter 8
非参数模型

Chapter 9
遗传学习

Chapter 10
强化学习

Chapter 11
优化方法

到目前为止，我们已经介绍了各种机器学习模型。这些模型基于不同的假设设计了不同的数据结构，因此需要不同的训练方法。这些各异的训练方法是各个领域的研究者长期工作的成果，具有很强的模型针对性，因而可实现高效学习。另一方面，这些方法又具有很强的共性，很多模型的训练方法基本原理都是相通的。例如不论是神经网络还是支持向量机，都会基于其目标函数的梯度来寻找模型的最优参数。我们可以对这些训练方法进行抽象，进而发现一些可应用于不同模型上的通用算法。这种抽象不仅可以让我们对现有算法有更深入理解，还可以对不同模型做对比印证，从而得到启发设计出更高效的训练方法。

原则上，如果我们可以将模型的训练目标形式化为一个参数化的目标函数$f(\mathbf{x})$，其中$\mathbf{x}$是模型中所有参数组成的向量，则任何模型训练问题都可归结为对$f(\mathbf{x})$的优化问题。依模型的具体形式不同，这一优化问题可能是连续的，也可能是离散的；可能是无约束的，也可能是带约束的；可能是凸的，也可能是非凸的。在第 **??**章中我们已经讨论过用演化学习、群体学习、随机优化等方法实现对$f(\mathbf{x})$的优化。然而，这些基于采样的优化方法效率低，收敛性无法保证，只能作为应对复杂模型的后备方法。特别是当模型中存在离散变量时，或推理复杂度非常高时，这种方法更为有效。对大多数模型，$f(\mathbf{x})$是连续甚至可微的，这时基于推理的优化方法更有效率。本章将讨论几种基于推理的优化算法，这些算法几乎在所有机器学习任务上被高频使用。值得说明的是，我们将讨论的这些算法都有成熟的工具包可供机器学习研究者直接使用，因此并不需要我们付出太多努力去研究其中的细节；然而，理解这些方法的基本思路、适用范围、对比优势，对合理使用这些方法却至关重要。

11.1 函数优化

11.1.1 优化问题定义

优化问题是应用数学的一个分支，被广泛应用于自然科学、工程、经济和工业生产中。从数学的角度来看，优化就是在满足一定限制条件的前提下，对目标函数进行最小化或者最大化。记$f(\mathbf{x})$为目标函数（我们仅考虑标量函数情况，这也是机器学习中最常用到的优化任务），$\mathbf{x}$为所有变量组成的向量（对应机器学习任务中的模型参数），则优化问题可以表示成如下的数学形式：

$$\min_{\mathbf{x}\in R^n} f(\mathbf{x})$$

$$c_i(\mathbf{x}) = 0,\ i \in \mathscr{E}$$
$$c_i(\mathbf{x}) \geq 0,\ i \in \mathscr{I},$$

其中，$\{c_i(\mathbf{x}) = 0; i \in \mathscr{E}\}$是等式约束条件，$\{c_i(\mathbf{x}) \geq 0, i \in \mathscr{I}\}$是不等约束条件，$\mathscr{E}$和$\mathscr{I}$分别是等式约束集和不等式约束集。上式形式上是一个带约束的优化问题，无约束优化问题可以认为是$\mathscr{E} = \mathscr{I} = \emptyset$的特例。

以一个二元目标函数为例，设：

$$f(\mathbf{x}) = \min_{x_1,x_2}(x_1-2)^2 + (x_2-1)^2$$

$$x_2 - x_1^2 \geq 0$$
$$-x_1 - x_2 \geq 2.$$

该优化问题如图 11.1 所示，其中两个约束边界的交点$\mathbf{x}^*$是最优解。

11.1.2 优化问题分类

依$f(\mathbf{x})$的性质和约束条件的特点，我们可以将优化任务简单分为如下几类：

- 离散优化和连续优化。如果$f(\mathbf{x})$中的变量$\mathbf{x}$只能取值一些离散的点，我们称优化任务为离散优化；如果$\mathbf{x}$可以在一个区域内连续取值，则称该任务为连续优化。离散优化中的主要问题是当$\mathbf{x}$的维度过大时会产生组合爆炸，

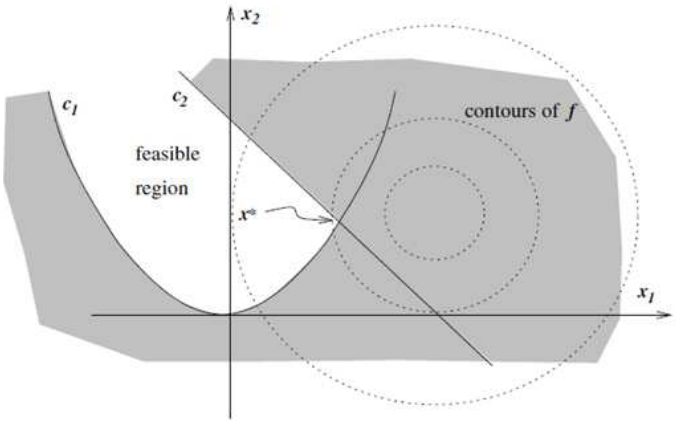

Fig. 11.1 一个带约束的二阶函数优化问题。$f(\mathbf{x}) = \min_{x_1,x_2}(x_1-2)^2+(x_2-1)^2$是待优化函数，其等值线表示为一组同心圆；$c_1 : x_2 - x_1^2 \geq 0$和$c_2 : -x_1 - x_2 \geq 2$ 是两个不等约束。图中的阴影区域是由这两个约束限制的非法区域。可见，$\mathbf{x}^*$是满足两个约束且使$f(\mathbf{x})$ 取最小值的最优解。

因此一般采用采样法求解。连续优化依赖$f(\mathbf{x})$的几何性质，特别是梯度和曲率信息对$f(\mathbf{x})$的优化值进行迭代搜索。**本章仅讨论连续优化问题，绝大多数机器学习算法属于这一类**。

- 无约束优化和带约束优化。如前所述，优化任务中如果不带约束条件，称为无约束优化，否则称为带约束优化。这两种优化任务有天然联系，一般来说我们希望将带约束优化通过某种变换转化成无约束优化，再依无约束优化方法进行求解。本章首先讨论无约束优化问题，然后讨论两种典型的带约束优化方法：一阶规划（Linear Programming）和二阶规划（Quadratic Programming），最后扩展到一般带约束优化问题。
- 全局优化和局部优化。全局优化是指找到$f(\mathbf{x})$的全局最优点，这在绝大多数情况下是很难实现的。这是因为机器学习任务中的目标函数$f(\mathbf{x})$一般有多个极值点，且变量$\mathbf{x}$维度非常高，很难从众多极值点中找到全局最优解。因此，绝大多数优化算法只关注局部优化。**本章我们主要讨论局部优化算法**。
- 凸优化和非凸优化。如果$f(\mathbf{x})$只有一个极值点，则称$f(\mathbf{x})$为凸函数（依极值是最大值还是最小值，也有可能是凹函数）。对凸函数的优化问题称为凸优化，否则为非凸优化。因为凸函数只有一个极值点，因此局部最优解即是全局最优解。绝大多数任务中$f(\mathbf{x})$是非凸的，因此凸优化方法无法直

接应用。然而，我们总可以在当前解的邻域内找到一个凸函数$m(\mathbf{x})$来近似在该点处的$f(\mathbf{x})$，从而利用凸优化方法求解。事实上这是很多重要优化方法的基本思路，如SGD，Newton，Quasi-Newton, SQP等。

在本章中，我们将讨论范围限定在连续的、非凸的、局部的优化任务。在不特别指明时，我们将假设目标函数具有任意阶导数，且这些导数是连续的。特别的，我们假设优化任务是寻找$f(\mathbf{x})$的**局部最小值**。局部最大值的优化任务可通过求其目标函数的负值转化成局部最小值优化任务，即：

$$\min_{\mathbf{x}} f(\mathbf{x}) = \max_{\mathbf{x}} \{-f(\mathbf{x})\}$$

11.1.3 基础定理

我们首先介绍几条在函数优化任务中经常用到的基础定理。这些定理是我们后续讨论所有优化算法的基础。这些定理的证明可从任何一本微积分教科书中查到 [9]。

泰勒定理 设$f(\mathbf{x})$是$R^n \to R$的连续可微函数，$\mathbf{x}_0 \in R^n$为其定义域上的任意一点，则总有$t \in (0,1)$使得下式成立：

$$f(\mathbf{x}_0 + \mathbf{p}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0 + t\mathbf{p})^{\mathrm{T}}\mathbf{p};$$

其中$\nabla f(\mathbf{x}_k) = \nabla f|_{\mathbf{x}_k} \in R^n$是$f(\mathbf{x})$在$\mathbf{x}_k$点的梯度。并且：

$$\nabla f(\mathbf{x}_0 + \mathbf{p}) = \nabla f(\mathbf{x}_0) + \int_0^1 \nabla^2 f(\mathbf{x}_0 + t\mathbf{p})\mathbf{p}dt$$

其中$\nabla^2 f(\mathbf{x}_k) = \nabla^2 f|_{\mathbf{x}_k} \in R^{n\times n}$是$f(\mathbf{x})$在$\mathbf{x}_k$点的Hessian矩阵。进而，如果$f(\mathbf{x})$二阶连续可微的，则有：

$$f(\mathbf{x}_0 + \mathbf{p}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^{\mathrm{T}}\mathbf{p} + \frac{1}{2}\mathbf{p}^T\nabla^2 f(\mathbf{x}_0 + t\mathbf{p})\mathbf{p}.$$

一阶必要条件 如果目标函数$f(\mathbf{x})$在$\mathbf{x}^*$的某个邻域内是一阶连续可微的，那么$\mathbf{x}^*$是局部极值的必要条件是$\nabla f(\mathbf{x}^*) = 0$。

二阶充要条件 如果目标函数$f(\mathbf{x})$在$\mathbf{x}^*$的某个领域内是二阶连续可微的，那么$\mathbf{x}^*$是局部极小值的充分必要条件是$\nabla f(\mathbf{x}^*) = 0$且$\nabla^2 f(\mathbf{x}^*)$是正定的。

11.2 无约束优化问题

无约束优化问题不考虑约束条件，对$f(\mathbf{x})$直接优化。对大多数实际问题，这一优化是没有闭式解的，因此大部分算法采取的策略是给定一个初始值$\mathbf{x}_0$，然后通过生成一个迭代序列$\mathbf{x}_1, \mathbf{x}_2, ...$来逼近局部最优解。大部分迭代算法可分为两类：线性搜索（Line Search）和置信域优化（Trust Region）。线性搜索首先确定一个合理的搜索方向，再基于这一方向搜索最优解；置信域优化首先设计一个局部近似函数$m(\mathbf{x})$，使该函数在当前解的某个邻域（称为置信域）内以足够的精度近似$f(\mathbf{x})$。对$m(\mathbf{x})$在该置信域内做优化即得到使$f(\mathbf{x})$更小的优化解。

11.2.1 线性搜索

在线性搜索中，首先从当前位置$\mathbf{x}_k$确定一个方向$\mathbf{p}_k$，使得目标函数在这个方向上有更小的函数值，然后在这个方向上进行搜索，确定步长α_k，使得依$\mathbf{p}_k$方向走过α_k 后得到的$\mathbf{x}_{k+1}$满足$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k)$。线性搜索的迭代过程可形式化如下：

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k. \tag{11.1}$$

11.2.1.1 梯度下降

梯度下降法（Gradient Descend，GD）是最简单，也是应用最广泛的线性搜索算法。依泰勒定理的一阶近似可知：

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) = f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) - f(\mathbf{x}_k) \approx \alpha_k \mathbf{p}_k^{\mathrm{T}} \nabla f_k,$$

其中我们将$\nabla f(\mathbf{x}_k)$简写为f_k。由上式可见，在一阶近似假设下，对一个确定的α_k，当取$\mathbf{p}_k$与∇f_k同方向时目标函数值的下降最大。这一结论是梯度下降法的基础。在这一方法中，首先确定$f(\mathbf{x})$在当前点$\mathbf{x}_k$ 的梯度，再沿梯度方向搜索最优步长α_k，并将优化解更新为$\mathbf{x}_k + \alpha_k \mathbf{p}_k$。

如果有足够的计算资源，可用较小步长沿$\mathbf{p}_k$方向尝试计算$f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)$，直到找到使该目标函数最小的$\alpha_k$为止。这种搜索一般比较耗时，因此绝大部分GD算法都会人为确定一个α_k。最简单的方法是设计一个随迭代递减的函

数（如$\alpha_k = 1/k$），在搜索初期选择较大的步长以加快搜索速度，当搜索逐渐收敛时减小步长以确定精细解。也可以保持某一α_k的值直到$f(\mathbf{x}_{k+1})$的取值开始上升，说明进入局部收敛区域，此时对α_k进行调整（如减半）以适应收敛要求。

在实际计算中，对于一些非常复杂的问题，计算目标函数的梯度并不容易。一种方法是对梯度做近似计算，只要得到的方向是使目标函数值下降的方向即可。如图 11.2所示，即使$\mathbf{p}_k$与∇f_k不完全同向，依此方向做线性搜索依然可以降低目标函数的值。

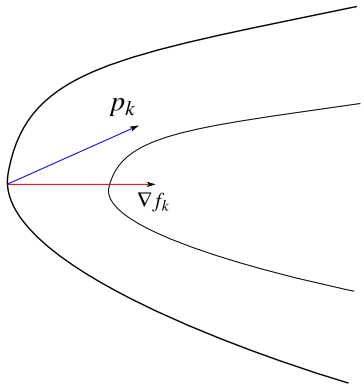

Fig. 11.2 非梯度方向的线性搜索。仅管$\mathbf{p}_k$ （蓝色单位向量）与梯度∇f_k（红色向量）方向不完全相同，沿$\mathbf{p}_k$方向搜索依然可使$f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$。

11.2.1.2 牛顿法

牛顿法（Newton Method）是另一种常用的线性搜索算法，这一方法的主要特点是可以利用目标函数的二阶曲率信息自动设置搜索的步长。依目标函数$f(\mathbf{x})$的二次泰勒展开式：

$$f(\mathbf{x}_k + \mathbf{p}) \approx f(\mathbf{x}_k) + \mathbf{p}_k^T \nabla f_k + \frac{1}{2}\mathbf{p}^T \nabla^2 f_k \mathbf{p}. \tag{11.2}$$

记上式右侧的近似函数为$m_k(\mathbf{p})$：

$$m_k(\mathbf{p}) = f(\mathbf{x}_k) + \mathbf{p}^T \nabla f_k + \frac{1}{2}\mathbf{p}^T \nabla^2 f_k \mathbf{p}.$$

注意$m_k(\mathbf{p})$是一个二阶凸函数，如果$\nabla^2 f_k$是正定的，则该近似函数的极小值满足：

$$\nabla m_k = \nabla f_k + \nabla^2 f_k \mathbf{p} = 0,$$

因而有：

$$\mathbf{p} = -(\nabla^2 f_k)^{-1} \nabla f_k. \tag{11.3}$$

由此我们得到牛顿法的迭代公式如下：

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k = \mathbf{x}_k - (\nabla^2 f_k)^{-1} \nabla f_k. \tag{11.4}$$

注意上式中我们并没有将位移$\mathbf{p}_k$分成方向和步长，而是直接计算出了它们的乘积，这极大避免了梯度下降算法中需要人为设定步长的困难。和递度下降法相比，牛顿法相当于在原来梯度方向上依曲率做了步长调整：由式 **??**可知，如果$f(\mathbf{x})$ 在$\mathbf{x}_k$点的某一方向曲率比较大，则$\nabla^2 f_k$在该方向的特征值比较大，因此在该方向的更新步长变小，从而避免在高曲率方向因步长过大导致的震荡；反之，如果在某一方向曲率比较小，则在该方向的步长变大，从而避免在低曲率方向因步长过小导致的更新缓慢。

注意的是，上述牛顿法的推导过程利用了目标函数的二阶近似。当这一近似误差不大时，利用公式 11.4 将得到很好的估计。事实上，如果$\nabla^2 f_k$是平滑的，$m_k(\mathbf{p})$与$f(\mathbf{x}_k + \mathbf{p})$ 的误差仅仅是$O(||\mathbf{p}||^3)$,因此当$\mathbf{v}$比较小时估计还是很精确的。但是由于公式 11.3中$\mathbf{p}$是直接算出来的，如果这一向量的模比较大则可能会产生较大误差。为防止这一误差带来的不精确性，可以对$\mathbf{p}$进行适当调节，如对其长度做一定限制，或在$\mathbf{p}$上乘以一个小量因子α_k。

11.2.1.3 拟牛顿法

牛顿法收敛速度快，可自动确定步长，因而是理想的优化方法。这一方法的优势在于利用于$f(\mathbf{x})$的二阶信息，即Hessian矩阵$\nabla^2 f_k$。然而，计算这一矩阵对大多数任务来说都是非常困难的。为了充分利用二阶信息，同时又能减少计算量，研究者们提出了很多方法来近似Hessian矩阵，这些方法称为拟牛顿法。

根据泰勒定理可知:

$$\nabla f(\mathbf{x}_{k+1}) = \nabla f(\mathbf{x}_k + \mathbf{p}_k) = \nabla f(\mathbf{x}_k) + \nabla^2 f(\mathbf{x}_k)\mathbf{p}_k + \int_0^1 [\nabla^2 f(\mathbf{x}_k + t\mathbf{p}_k) - \nabla^2 f(\mathbf{x}_k)]\mathbf{p}_k dt,$$

其中:

$$\int_0^1 [\nabla^2 f(\mathbf{x}_k + t\mathbf{p}_k) - \nabla^2 f(\mathbf{x}_k)]\mathbf{p}_k dt = o(||\mathbf{p}_k||_1)$$

所以

$$\nabla f_{k+1} = \nabla f_k + \nabla^2 f_k \mathbf{p}_k + o(||\mathbf{p}_k||_1),$$

因而有：

$$\nabla^2 f_k \mathbf{p}_k \approx \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k),$$

注意$\nabla^2 f_k$是$\mathbf{x}_k$处$f(\mathbf{x})$的Hessian矩阵，记为$\mathbf{H}_k$。上式表明$\mathbf{H}_k$与前后两次递度的差有直接关系。由此，我们希望找到一个$\mathbf{H}_k$的近似矩阵$\mathbf{B}_{k+1}$，使其具有和上式同样的关系：

$$\mathbf{H}_k \mathbf{p}_k \approx \mathbf{B}_{k+1}\mathbf{p}_k = \mathbf{y}_k,$$

其中

$$\mathbf{p}_k = \mathbf{x}_{k+1} - \mathbf{x}_k; \quad \mathbf{y}_k = \nabla f_{k+1} - \nabla f_k.$$

一般来说，为简化计算需要对$\mathbf{B}_k$做更多限制。首先一般假设$\mathbf{B}_k$是对称的，因为$\mathbf{H}_k$本身也是对称的。另外，通常假设$\mathbf{B}_{k+1} - \mathbf{B}_k$是低阶的，因为我们希望Hessian矩阵在相邻两次迭代中的变动不大。

一种$\mathbf{B}_k$的更新方法称为Symmetric-Rank-One (SR1) 更新，其中$\mathbf{B}_{k+1} - \mathbf{B}_k$是一阶的，更新公式为：

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{(\mathbf{y}_k - \mathbf{B}_k\mathbf{p}_k)(\mathbf{y}_k - \mathbf{B}_k\mathbf{p}_k)^T}{(\mathbf{y}_k - \mathbf{B}_k\mathbf{p}_k)^T\mathbf{p}_k}.$$

另一种著名的更新方法称为BFGS更新, 由其四位发明者的名字（Broyden, Fletcher, Goldfarb, and Shanno）命名。在这一更新算法中，$\mathbf{B}_{k+1} - \mathbf{B}_k$是二阶的，且只要初始$\mathbf{B}_0$是正定的，即可保证所有$\mathbf{B}_k$都是正定的。BFGS的更新公式为:

$$\mathbf{B}_{k+1} = \mathbf{B}_k - \frac{\mathbf{B}_k\mathbf{p}_k\mathbf{p}_k^T\mathbf{B}_k}{\mathbf{p}_k^T\mathbf{B}_k\mathbf{p}_k} + \frac{\mathbf{y}_k\mathbf{y}_k^T}{\mathbf{y}_k^T\mathbf{p}_k}.$$

不论是SR1或BFGS，得到$\mathbf{H}_k$的近似$\mathbf{B}_k$后，即可依牛顿法得到如下拟牛顿优化算法的迭代公式：

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{B}_k^{-1} f(\mathbf{x}_k).$$

上述迭代公式事实上需要的是$\mathbf{B}_k^{-1}$而非$\mathbf{B}_k$。记$\mathbf{V}_k = \mathbf{B}_k^{-1}$，可以调整SR1和BFGS 的迭代公式直接计算$\mathbf{V}_k$。对于SR1，$\mathbf{V}_k$的更新公式如下：

$$\mathbf{V}_{k+1} = \mathbf{V}_k + \frac{(\mathbf{p}_k - \mathbf{V}_k\mathbf{y}_k)(\mathbf{p}_k - \mathbf{V}_k\mathbf{y}_k)^T}{(\mathbf{p}_k - \mathbf{B}_k\mathbf{y}_k)^T\mathbf{y}_k},$$

对于BFGS，$\mathbf{V}_k$更新公式如下：

$$\mathbf{V}_{k+1} = (\mathbf{I} - \rho_k\mathbf{p}_k\mathbf{y}_k^T)\mathbf{V}_k(\mathbf{I} - \rho_k\mathbf{y}_k\mathbf{p}_k^T) + \rho_k\mathbf{p}_k\mathbf{p}_k^T,$$

其中:

$$\rho_k = \frac{1}{\mathbf{y}_k^T\mathbf{p}_k}.$$

11.2.1.4 共轭梯度法

共轭梯度法最初提出来是为了求解线性方程$\mathbf{Ax} = \mathbf{b}$ 所确定的线性系统，其中系数矩阵$\mathbf{A} \in R^{n\times n}$是一个对称正定的矩阵。显然，求解该线性方程组的问题等价于求如下二价凸函数的最小值问题：

$$m(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{Ax} - \mathbf{b}^T\mathbf{x}. \tag{11.5}$$

对于任意一个目标函数$f(\mathbf{x})$，在$\mathbf{x}_k$点取如下二阶近似:

$$m_k(\mathbf{p}) = f(\mathbf{x}_k) + \mathbf{p}^T\nabla f_k + \frac{1}{2}\mathbf{p}^T\mathbf{H}_k\mathbf{p}.$$

上式和式 11.5具有相同形式，因此可通过解线性方程$\mathbf{H}_k\mathbf{p} = -\nabla f_k$得到$\mathbf{p}$ 的最优解作为$\mathbf{p}_k$。这是共轭梯度法用于优化问题的基本原理。

不失一般性，以线性方程$\mathbf{Ax} = \mathbf{b}$讨论共轭梯度算法。假设$\mathbf{A}$很大，以至无法用高斯消元法等标准算法直接求解。共轭递度下降法设计了一个迭代过程，从$\mathbf{x}_0$开始，每次迭代生成一个位移方向$\mathbf{d}_k$，该方向与所有已经生成的位移方向具有如下共轭关系：

$$\mathbf{d}_k^T\mathbf{A}\mathbf{d}_j = 0, \quad j < k.$$

理论上，当搜索n步时必然会找到$\mathbf{Ax}=\mathbf{b}$的解，其中n是矩阵$\mathbf{A}$的列数；实际实现时，往往几轮迭代后就可以得到很好的结果。共轭梯度算法的细节由算法 1给出。

Input: $\mathbf{H}, \mathbf{b}$;
Initialization: $\mathbf{x}_0 = 0; \mathbf{r}_0 = b; k = 0$;
for *k:=1 to N* **do**
  if *k==1* **then**
    $\mathbf{d}_k = \mathbf{r}_0$;
  end
  else
    $\mathbf{d}_k = \mathbf{r}_{k-1} + \frac{\mathbf{r}_{k-1}^T \mathbf{r}_{k-1}}{\mathbf{r}_{k-2}^T \mathbf{r}_{k-2}} \mathbf{d}_{k-1}$;
  end
  $\alpha_k = \frac{\mathbf{r}_{k-1}^T \mathbf{r}_{k-1}}{\mathbf{d}_k^T \mathbf{A} \mathbf{d}_k}$;
  $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_k \mathbf{d}_k$;
  $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_k \mathbf{A} \mathbf{d}_k$;
  if $||\mathbf{r}_k|| < \delta$ **then**
    Break;
  end
end
Output: $\mathbf{x}_k$;

Algorithm 1: 对线性方程$\mathbf{Ax}=\mathbf{b}$求解的共轭梯度算法。其中$\mathbf{d}_k$为第k步的搜索方向，$\mathbf{r}_k$为第k步的残差，α_k 为第k 步步长，$\mathbf{x}_k$ 为第k 步的优化结果。

上述共轭梯度法应用于大规模优化问题时的一个显著优点是，在计算第k步的步长和残差时，虽然都会用到矩阵$\mathbf{A}$ ，但事实上只需计算向量$\mathbf{Ad}_k$。对于$f(\mathbf{x})$的优化而言，$\mathbf{A}$事实上对应的是很难求解的Hessian矩阵，但$\mathbf{Ad}_k$却可以通过数值法求出，因此极大降低了计算开销。一般来说，共轭梯度法比梯度下降法更为有效，也更容易计算。虽然它不像牛顿法和拟牛顿法具有那么快的收敛速度，但共轭梯度法不需要计算Hessian矩阵，因此是很多大型问题的理想优化算法。图 11.3比较了共轭梯度法与标准梯度下降法的搜索路径，可以看到共轭梯度法具有更好的收敛性质。

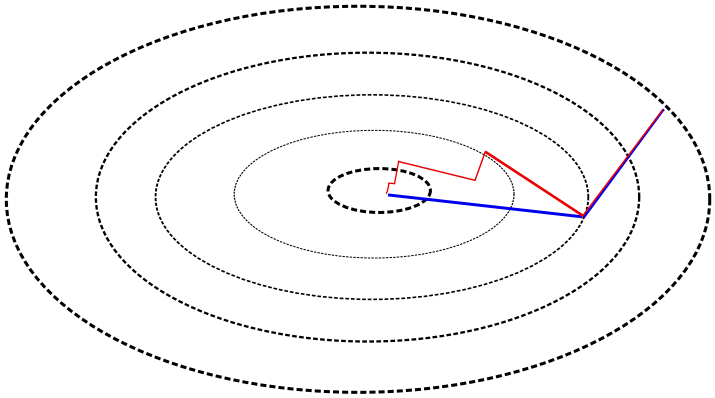

Fig. 11.3 共轭梯度法与梯度下降法搜索路径比较。红色折线为标准梯度下降法，蓝线为共轭梯度法。可以看到共轭梯度法具有更有效的收敛路径。

11.2.2 置信域优化

光滑函数上无约束优化问题的另一种方法是置信域优化。在这一方法中，首先确定当前解$\mathbf{x}_k$的一个置信域，在这个区域内$f(\mathbf{x}_k+\mathbf{p})$可被某个函数$m_k(\mathbf{p})$近似。置信域确定后，对近似函数$m_k(\mathbf{p})$在该域内进行优化，用得到的优化值近似原函数的优化值。该方法形式化如下：

$$\mathbf{p}^*(\Delta) = \arg\min_{\mathbf{p}} \ m_k(\mathbf{p}) \ \ s.t. \ \ ||\mathbf{p}|| \leq \Delta,$$

其中Δ为置信域的并径，$\mathbf{p}^*(\Delta)$为半径为Δ的置信域内近似函数$m_k(\mathbf{p})$的优化解。

如果取一阶近似，依泰勒定理，近似函数：

$$m_k^1(\mathbf{p}) = f(\mathbf{x}_k) + \mathbf{p}^T \nabla f_k \tag{11.6}$$

如果取二阶近似，则有近似函数：

$$m_k^2(\mathbf{p}) = f(\mathbf{x}_k) + \mathbf{p}^T \nabla f_k + \frac{1}{2}\mathbf{p}^T \mathbf{B}_k \mathbf{p}. \tag{11.7}$$

其中$\mathbf{B}_k$是在$f(\mathbf{x})$在$\mathbf{x}_k$处Hessian矩阵的近似。

11.2.2.1 Dogleg算法

置信区域优化将$f(\mathbf{x})$的无约束优化问题转化成近似函数$m_k(\mathbf{p})$的带约束优化问题。这一带约束问题可用任何方法求解，得到的最优解$\mathbf{p}^*(\Delta)$是置信域半径Δ的连续函数。如果我们仅考虑一阶近似$m_k^1(\mathbf{p})$和二阶近似$m_k^2(\mathbf{p})$两种近似函数，$\mathbf{p}^*(\Delta)$的形式相对简单。

$$f(\mathbf{x}_k+\mathbf{p}) \approx m_k^1(\mathbf{p}) = f(\mathbf{x}_k) + \mathbf{p}^T \nabla f_k$$

$$f(\mathbf{x}_k+\mathbf{p}) \approx m_k^2(\mathbf{p}) = f(\mathbf{x}_k) + \mathbf{p}^T \nabla f_k + \mathbf{p}^T B_k \mathbf{p}$$

其中B_k为在$\mathbf{x}_k$点$f(\mathbf{x})$的近似Hessian矩阵。

一阶近似优化点

对于一阶近似，易知对任意一个Δ，$\mathbf{g} = \frac{\nabla f_k}{||\nabla f_k||}$ 是使$m_k^1(\mathbf{p})$ 下降最快的方向，因此受限优化问题的最优解$\mathbf{p}^*(\Delta)$在$\mathbf{g}$所确定的直线上，且是$\mathbf{g}$与圆$||\mathbf{p}|| = \Delta$的交点。这意味着$\mathbf{p}^*(\Delta)$ 随着Δ的增大表现为沿着$\mathbf{g}$方向的直线，即$\mathbf{p}^*(\Delta) = -\Delta\mathbf{g}$。值得注意的是，只有当$\Delta$值比较小时，$m_k^1(\mathbf{p})$对$f(\mathbf{x}_k+\mathbf{p})$的近似才成立，这意味着一阶近似的优化点$\mathbf{p}^*(\Delta)$只有当$\Delta$比较小时才是对原函数$f(\mathbf{x})$的较好优化位移。

我们可以求$f(\mathbf{x})$在$\mathbf{g}$方向的最小值位置$\tilde{\mathbf{x}}$，如果在沿$\mathbf{g}$做线性搜索时超过这一位置，一阶近似函数$m_k^1(\mathbf{p})$ 将无法继续近似$f(\mathbf{x})$。这是因为当超过这一位置后，$f(\mathbf{x})$ 值将上升，而$m_k^1(\mathbf{p})$将继续下降。直接求$f(\mathbf{x})$ 在$\mathbf{g}$方向上的最小值点$\tilde{\mathbf{x}}$并不容易，我们利用$f(\mathbf{x})$的二阶近似$m_k^2(\mathbf{p})$来求这一最小值的近似值，即求$m_k^2(\mathbf{p})$在$\mathbf{g} = \nabla f_k$方向的极小值点，因而有如下关系：

$$\min_{\mathbf{x}} f(\mathbf{x}) \approx \min_{\alpha} m_k^2(\alpha\mathbf{g}) = \min_{\alpha}\{f_k + \alpha\mathbf{g}^T\mathbf{g} + \frac{\alpha^2}{2}\mathbf{g}^T\mathbf{B}_k\mathbf{g}\},$$

可得最小值点对应的α为:

$$\alpha = -\frac{\mathbf{g}^T\mathbf{g}}{\mathbf{g}^T\mathbf{B}_k\mathbf{g}},$$

由此得到在$\mathbf{g}$方向上$f(\mathbf{x})$的最小值点为：

$$\tilde{\mathbf{x}} \approx \mathbf{x}_k + \mathbf{p}^U,$$

其中：

$$\mathbf{p}^U = \alpha\mathbf{g} = -\frac{\mathbf{g}^T\mathbf{g}}{\mathbf{g}^T\mathbf{B}_k\mathbf{g}}\mathbf{g}.$$

对任何$\Delta \leq \alpha$的置信域，一阶近似函数$m_k^1(\mathbf{p})$在受限条件下的最小值点为：

$$\mathbf{p}^*(\Delta) = \frac{\Delta}{\alpha}\mathbf{p}^U = \tau\mathbf{p}^U$$

其中$\tau \leq 1$表示Δ与全局最小值点的步长α的比例。

二阶近似优化点

对于二阶近似函数$m_k^2(\mathbf{p})$，在没有Δ限制时，易知其最小值点对应的更新向量为$\mathbf{p}^B = -\mathbf{B}^{-1}\nabla f_k$。这说明当$\Delta \geq ||\mathbf{B}^{-1}\nabla f_k||$时，$\Delta$的限制并不起作用，此时带限制条件的最优结果$\mathbf{p}^*(\Delta)$即是$m_k^2(\mathbf{p})$在没有$\Delta$限制时的最优解$\mathbf{p}^B$。反之，如果$\Delta$过小，受限二阶优化得到的$\mathbf{p}^*(\Delta)$与无限制二阶优化的结果会相差较大。

Dogleg 优化轨迹

综上所述，当Δ比较小时，适合用一阶近似，当Δ比较大时，需要用二阶近似。一种思路是将二者结合起来，得到一种联合优化方法，称为Dogleg方法，如下式所示：

$$\mathbf{p}^*(\tau)\begin{cases}\tau\mathbf{p}^U & 0 \leq \tau \leq 1\\ \mathbf{p}^U + (\tau-1)(\mathbf{p}^B - \mathbf{p}^U) & 1 \leq \tau \leq 2\\ \mathbf{p}^B & \tau \geq 2.\end{cases}$$

上式中的τ即为我们前面定义的Δ与一阶近似中g方向上最优步长α的比例。由上式可见，当$\tau \leq 1$时，$\mathbf{p}^*(\tau)$ 取一阶近似在τ限制下的最优位移；当$1 \geq \tau \geq 2$ 时，取一阶近似和二阶近似最优位移的线性组合；当$\tau \geq 2$时（即$\Delta = 2\alpha$），只取二阶近似的最优位移即可。由此，对Δ的搜索转化为对差值参数τ 在$[0,2]$之间的搜索。图 11.4给出Dogleg 方法中$\mathbf{p}^*(\Delta)$的轨迹。

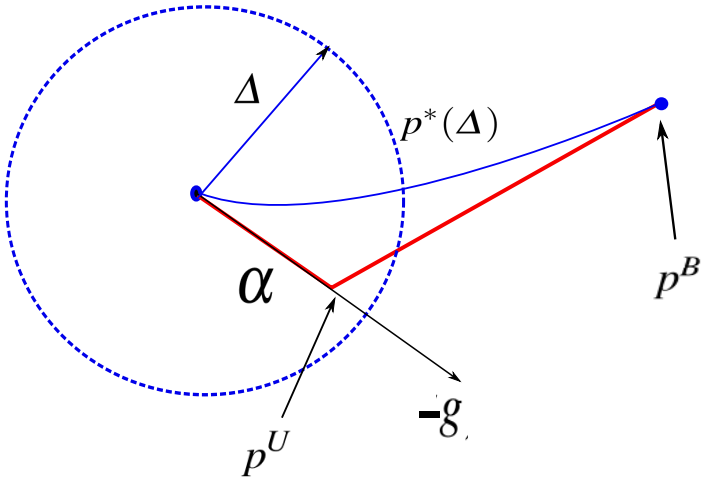

Fig. 11.4 Dogleg算法示意图。蓝色曲线为基于二阶近似$m_k^2(\mathbf{p})$的受限优化问题的优化点$\mathbf{p}^*(\Delta)$，红色折线为Dogleg方法得到的受限优化问题的优化点$\mathbf{p}^*(\tau)$。$\mathbf{g}$为$f(\mathbf{x})$的梯度下降方向，α为在该方向上以$m_k^2(\mathbf{p})$为目标函数的最优值所对应的Δ，$\mathbf{p}^U$为对应的最优点。$\mathbf{p}^B$为二阶近似$m_k^2(\mathbf{p})$在无条件限制下的最优点。

11.2.2.2 置信域调整

置信域算法需要确定在点$\mathbf{x}_k$处的置信域半径Δ_k，使得近似函数$m_k(\mathbf{p})$对目标函数$f(\mathbf{x})$ 在该区域内具有较好的近似能力。这一近似能力可通过如下比例因子检测：

$$\rho_k = \frac{f(\mathbf{x}_k) - f(\mathbf{x}_k + \mathbf{p}_k)}{m_k(\mathbf{0}) - m_k(\mathbf{p}_k)}. \tag{11.8}$$

显然，$m_k(\mathbf{p}_k)$与$f(\mathbf{x}_k)$偏离越远，ρ_k的值越小。基于此，可以逐渐调整置信区间的半径Δ_k，使得$m_k(\mathbf{p}_k)$与$f(\mathbf{x}_k)$保持较好的一致性。

具体来说，首先注意到近似函数上的差值$m_k(\mathbf{0}) - m_k(\mathbf{p}_k)$是非负的，所以，当$\rho_k$是负数时，说明$f(\mathbf{x}_k + \mathbf{p}_k))$比$f(\mathbf{x}_k)$大，因此需要拒绝$\mathbf{p}_k$。这同时意味着此时的置信域过大，近似函数无法在该区域内对$f(\mathbf{x})$做很好近似，因此需要减小Δ_k的值，再次计算优化的$\mathbf{p}_k$；如果ρ_k是正的但接近于零，本次迭代得到的$\mathbf{p}_k$可以保留，但需在下一次迭代中缩小Δ；如果ρ_k取值较大，说明当前置信域选择比较合理，可以在下次迭代时继续使用；如果当ρ_k的取值接近1，说明在当前置信域内近似函数$m_k(\mathbf{p})$对$f(\mathbf{x})$有很好的近似，这时我们可以在下一次迭代时进一步扩大置信域，以提高搜索效率。

11.2.2.3 线性搜索与置信域优化比较

线性搜索和置信域优化是两种解决无约束优化问题的两种基本方法。在线性搜索中，我们首先找到一个搜索方向$\mathbf{p}_k$，使得从当前取值点$\mathbf{x}_k$沿该方向搜索时目标函数下降，再确定在该方向的搜索步长，找到一个使$f(\mathbf{x})$下降最大的优化点。置信域优化是先确定一个置信域，在该置信域内计算近似函数的最小取值点。由于该近似函数在这一置信域内可以较好近似原目标函数，因此依该近似函数得到的最小取值点也以可代表原目标函数在$\mathbf{x}_k$附近的最小值点。总体来说，线性搜索是先方向后步长的搜索方法，置信域是先步长后方向的搜索方法。

这两种算法有密切联系：不管哪种方法，都强烈依赖原函数$f(\mathbf{x})$的梯度和曲率信息，设计一阶或二阶近似函数，并基于该近似函数的优化方向和步长来确定$f(\mathbf{x})$的下一个优化点。这两种方法也互相借鉴。例如在牛顿方法中，如果$||\mathbf{x}_{k+1}-\mathbf{x}_k||$过大，则有可能需要对步长进行控制，以防止二阶近似带来的过大误差，这事实上正是置信域的概念。

11.3 带约束优化问题

前面已经提到，带约束优化是在无约束优化基础上加入一组限制条件。这类问题一般可以表示成如下形式：

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad s.t. \begin{cases} c_i(\mathbf{x}) = 0, & i \in \mathscr{E} \\ c_i(\mathbf{x}) \geq 0, & i \in \mathscr{I}. \end{cases} \tag{11.9}$$

其中$f(\mathbf{x})$是目标函数，$\mathscr{E}$是等式约束条件集，$\mathscr{I}$ 是不等约束条件集。设这些约束函数$c_i(\cdot)$均是定义在R^n上的光滑函数，我们通常把满足所有约束条件的点$\mathbf{x}$组成的集合称为合法域，即：

$$\Omega = \{\mathbf{x} | c_i(\mathbf{x}) = 0,\ i \in \mathscr{E};\ \ c_i(\mathbf{x}) \geq 0, i \in \mathscr{I}\}. \tag{11.10}$$

因此，上述带约束的最优问题也可以简写成如下形式：

$$\min_{\mathbf{x} \in \Omega} f(\mathbf{x}). \tag{11.11}$$

11.3.1 拉格朗日乘子法

拉格朗日乘子法是解决带约束优化问题的基本方法。我们以等式约束来说明这一重要方法，并扩展到非等式约束上。以包含一个等式约束的优化问题为例：

$$\min f(\mathbf{x}) \ \ s.t. \ \ g(\mathbf{x}) = 0. \tag{11.12}$$

设上述带约束问题的最优值为a，则等值线$f(\mathbf{x}) = a$必然与曲线$g(\mathbf{x}) = 0$相切，如图 11.5所示。这是因为如果二者不相切，在二者的点$\mathbf{x}^*$处沿$g(\mathbf{x}) = 0$的切线方向做微小移动时，总会有沿$f(\mathbf{x})$法线方向的分量使得$f(x)$的取值发生变化，进而使$f(\mathbf{x}) < f(\mathbf{x}^*)$。$f(\mathbf{x}) = a$和$g(\mathbf{x}) = 0$这两条直线在$\mathbf{x}^*$点相切意味着在点$\mathbf{x}^*$，必然有这两条曲线的梯度在一条直线上（方向未必相同），即$\mathbf{x}^*$需满足如下条件：

$$\nabla f(\mathbf{x}^*) = \lambda \nabla g(\mathbf{x}^*). \tag{11.13}$$

注意上式是问题 11.12的必要条件，而非充分条件。事实上，对任何一个确定的目标函数$f(\mathbf{x})$ 和限制条件$g(\mathbf{x}) = a$，都在切点$\mathbf{x}^*$有一个确定的λ^*，只有基于该λ^*值，式 11.13 才成立。因为不同约束条件对应不同的λ^*，引入约束条件本身将和式 11.13一起确定λ^*的取值，即：

$$\nabla f(\mathbf{x}^*) = \lambda^* \nabla g(\mathbf{x}^*) \tag{11.14}$$

$$g(\mathbf{x}^*) = 0. \tag{11.15}$$

容易验证，11.14-11.15式正是如下无约束优化问题取优化点$(\mathbf{x}^*, \lambda^*)$的一阶必要条件：

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda g(\mathbf{x}).$$

这一结果意味着一个带约束问题可以转化为无约束问题，该无约束问题的优化解的部分变量即为原带约束问题的解。这一方法称为拉格朗日乘子法，其中参数λ称为拉格朗日乘子，$L(\mathbf{x}, \lambda)$称为拉格朗日目标函数。

将上述等式约束下的拉格朗日乘子法扩展到包含不等约束的优化任务。以如下包含单一不等约束的优化问题为例：

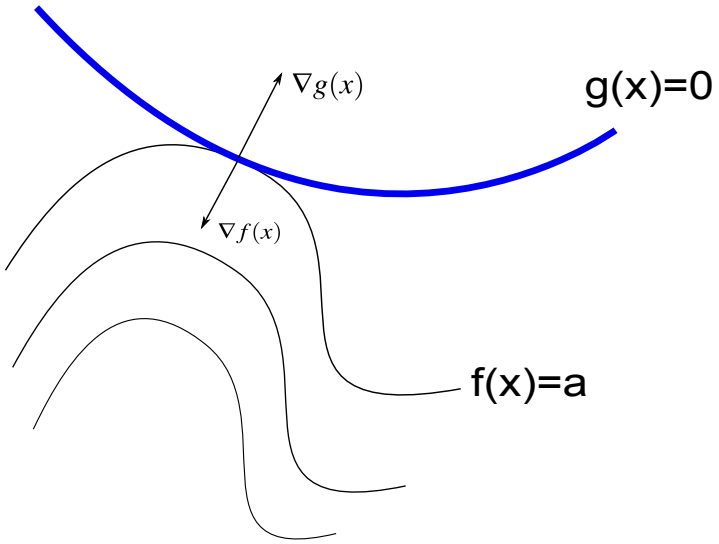

Fig. 11.5 带等式约束的拉格朗日乘子法。$f(\mathbf{x})$是待优化函数，$f(\mathbf{x})=a$是$f(\mathbf{x})$的一条等值线，$g(\mathbf{x})=0$是等式约束所对应的曲线。$f(\mathbf{x})=a$与$g(\mathbf{x})=0$相切，在切点处有$\nabla f(\mathbf{x})=\lambda\nabla g(\mathbf{x})$。

$$\min f(\mathbf{x}) \;\; s.t. \;\; h(\mathbf{x}) \geq 0. \tag{11.16}$$

这一优化问题的解*vecx**有如下两种情况（见图 11.6）：(1). 如果$\mathbf{x}^*$在曲线$h(\mathbf{x})=0$上，即回归到等式约束条件，因而有: $\nabla f(\mathbf{x}^*)=\lambda^*\nabla h(\mathbf{x}^*); \lambda^*>0$。注意这时曲线$f(\mathbf{x})=a$和$h(\mathbf{x})=0$ 在切点$\mathbf{x}^*$处的梯度方向必然是相同的，否则必然有满足$h(\mathbf{x})>0$的点$\mathbf{x}$使得$f(\mathbf{x})<f(\mathbf{x}^*)$。2. 如果$\mathbf{x}^*$在曲线$h(\mathbf{x})=0$以内，即$h(\mathbf{x}^*)>0$，此时$h(\mathbf{x})$的限制不起作用，带约束的优化问题与不带约束的优化问题的解是一致的。

我们将上述两种情况总结为统一的格拉朗日目标函数形式，可发现式11.16所示的带约束优化问题等价于如下无约束优化问题：

$$L(\mathbf{x},\lambda)=f(\mathbf{x})-\lambda h(\mathbf{x})$$

其优化值$(\mathbf{x}^*,\lambda^*)$需满足如下条件(称为Karush-Kuhn-Tucker 条件)：

$$\begin{aligned}
&\nabla(\mathbf{x}^*)=\lambda^* h(\mathbf{x}^*)\\
&h(\mathbf{x}^*)\geq 0\\
&\lambda^*\geq 0,\\
&\lambda^* h(\mathbf{x}^*)=0.
\end{aligned} \tag{11.17}$$

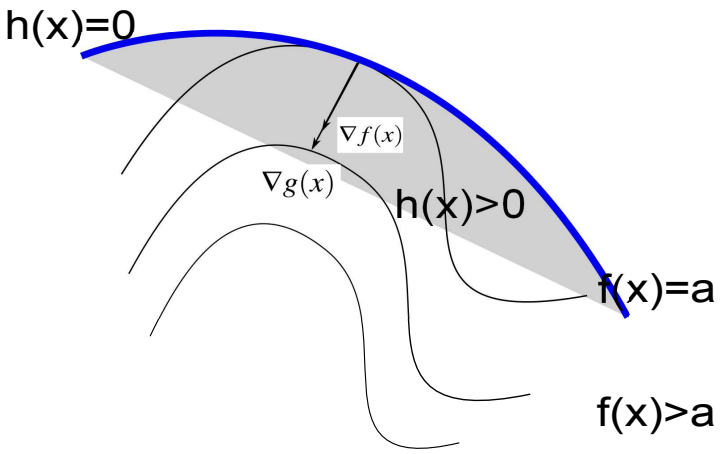

Fig. 11.6 带不等约束的拉格朗日乘子法。$f(\mathbf{x})$是待优化函数，$f(\mathbf{x})=a$是$f(\mathbf{x})$的一条等值线，$h(\mathbf{x})\geq 0$是等非等约束所对应的合法区域。如果最优点$\mathbf{x}^*$在曲线$h(\mathbf{x})=0$上，则$f(\mathbf{x})=a$与$h(\mathbf{x})=0$在$\mathbf{x}^*$相切，且梯度方向相同。如果$\mathbf{x}^*$不在$h(\mathbf{x})=0$上，则转化为无约束优化问题。

换句话说，$h(\mathbf{x}^*)$和$\boldsymbol{\lambda}^*$只要有一个不是零，则另一个必然为零。这一KKT条件在我们讨论SVM时有重要意义，正是因为$h(\mathbf{x}^*)>0$时必有$\boldsymbol{\lambda}^*=0$，才使得非支持向量在模型中的权重被降为零，从而大大减小了模型复杂度（见第5章）。

总结起来，一个如式11.9所示的带约束优化问题等价于如下无约束优化问题：

$$L(\mathbf{x},\boldsymbol{\lambda})=f(\mathbf{x})-\sum_{i\in\mathscr{E}}\lambda_i c_i(\mathbf{x})-\sum_{i\in\mathscr{I}}\lambda_i c_i(\mathbf{x}),$$

并满足如下KKT条件：

$$\begin{aligned}&\lambda_i^*\geq 0 && \forall i\in\mathscr{I}\\ &c_i(\mathbf{x}^*)\geq 0 && \forall i\in\mathscr{I}\\ &\lambda_i^* c_i(\mathbf{x}^*)=0 && \forall i\in\mathscr{I}.\end{aligned} \tag{11.18}$$

11.3.2 对偶问题

如果$f(\mathbf{x})$和$\{-c_i(\mathbf{x})\}$都是凸函数，则$L(\mathbf{x},\boldsymbol{\lambda})$ 也是凸的。这类问题可以用对偶解法。记优化问题为:

$$\min_{\mathbf{x}} f(\mathbf{x}) \;\; s.t. \;\; c_i(\mathbf{x}) \geq 0, i \in \mathscr{I}. \tag{11.19}$$

记$\mathbf{c} = (c_1, ..., c_n)^T$，将上述优化问题写成拉格朗日目标函数形式，有：

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{x}). \tag{11.20}$$

定义如下函数：

$$q(\boldsymbol{\lambda}) = \inf_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda}) \tag{11.21}$$

则原问题 的对偶问题表示为:

$$\max_{\boldsymbol{\lambda}} q(\boldsymbol{\lambda}) \;\; s.t. \;\; \boldsymbol{\lambda} \geq \mathbf{0}. \tag{11.22}$$

可以证明，在特定条件下，上述对偶问题得到的最优值与原问题11.3.2得到的最优值是一致的，且对偶问题的最优解$\boldsymbol{\lambda}^*$代入式 11.20中，最小化$L(\mathbf{x}, \boldsymbol{\lambda}^*)$ 得到的$\mathbf{x}^*$正是原问题 11.3.2的解（[8], Theorem 12.12）。换句话说，$\mathbf{x}^*$和$\boldsymbol{\lambda}^*$是一对对偶解，分别解原问题 11.3.2和对偶问题 ，二者统一到拉格朗日乘子目标函数 11.20，$(\mathbf{x}^*, \boldsymbol{\lambda}^*)$是该目标函数的联合最优解。注意上述结论依赖$f(\mathbf{x})$和$\{-c_i(\mathbf{x})\}$的凸函数性质。图 11.7给出上述对偶问题的示意图。对偶问题在很多机器学习任务中具有重要意义。一方面它可提供效率更高的解法，另一方面可以揭示算法本身的深层属性。下一节要讨论的线性规划问题即应用了这种对偶解法。

以一个例子来说明对偶问题。设如下带一个不等约束的优化问题：

$$\min_{x_1, x_2} f(x_1, x_2) = \min_{x_1, x_2} 0.5(x_1^2 + x_2^2) \;\; s.t. \;\; x_1 - 1 \geq 0.$$

依拉格朗日乘子公式：

$$L(x_1, x_2, \lambda) = 0.5(x_1^2 + x_2^2) - \lambda(x_1 - 1).$$

依式 11.21，$q(\lambda)$是给定一个λ后$L(x_1, x_2, \lambda)$的最小值。这一优化任务可以通过求$L(x_1, x_2, \lambda)$在(x_1, x_2)的驻点得到：

$$\frac{\partial L(x_1, x_2, \lambda)}{\partial x_1} = 0; \;\; \frac{\partial L(x_1, x_2, \lambda)}{\partial x_2} = 0.$$

整理可得：

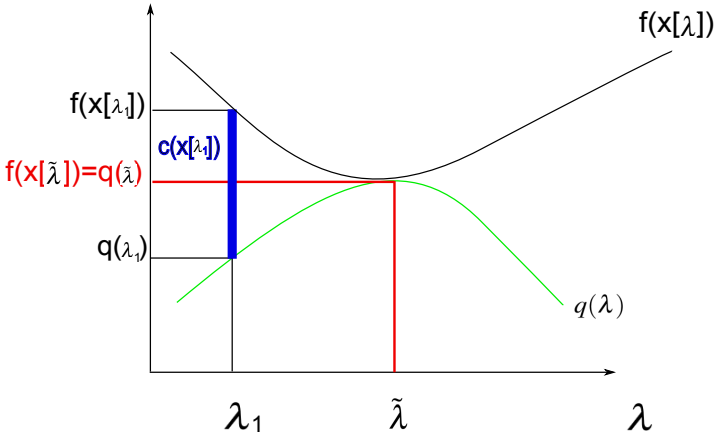

Fig. 11.7 包含一个约束的对偶问题。$q(\lambda)$和$f(\mathbf{x})$是原问题和对偶问题的目标函数。$q(\lambda)$的最大值等于$f(\mathbf{x})$的最小值。对任意一点λ，$q(\lambda)$包括两部分：$f(\mathbf{x}[\lambda])$和$c(\mathbf{x}[\lambda])$，其中$\mathbf{x}[\lambda]$是使式 11.21达到下界的$\mathbf{x}$。这两部分相减得到$q(\lambda)$。在$q(\lambda)$的最大值点λ^* 有$c(\mathbf{x}[\lambda^*]) = 0$, 此时$f(\mathbf{x}[\lambda^*]) = q(\lambda^*)$。

$$x_1^* - \lambda = 0;\ \ x_2^* = 0.$$

将上式代入$L(x_1, x_2, \lambda)$，有:

$$q(\lambda) = 0.5(\lambda^2 + 0) - \lambda(\lambda - 1) = -0.5\lambda^2 + \lambda.$$

如果我们对上式优化，可得:

$$\lambda^* = \arg\max_{\lambda \geq 0} q(\lambda) = 1$$

$$q(\lambda^*) = 0.5.$$

如果我们将λ^*代入$L(x_1, x_2, \lambda)$，并求最大解，有：

$$x_1^* = 1,\ \ x_2^* = 0.$$

将上式代入原优化任务，可得：

$$f(x_1^*, x_2^*) = 0.5 = q(\lambda^*).$$

上述对偶问题表示中$q(\boldsymbol{\lambda})$包含极小函数$\inf_{\mathbf{x}}(\cdot)$，形式比较复杂。另一种更直观的对偶问题表示称为Wolfe对偶表示，形式化如下：

$$\max_{\mathbf{x},\boldsymbol{\lambda}} L(\mathbf{x},\boldsymbol{\lambda}) \quad s.t.\ \nabla_{\mathbf{x}} L(\mathbf{x},\boldsymbol{\lambda}) = \mathbf{0},\ \boldsymbol{\lambda} \geq \mathbf{0}.$$

注意上式中的优化任务是对拉格朗日目标函数的最大化。可以证明如果$\mathbf{x}^*, \boldsymbol{\lambda}^*$是原问题的解，则他们也是Wolfe对偶问题的解。

11.3.3 线性规划

线性规划（Linear Programming, LP）是最简单的带约束优化任务，其目标函数和约束条件都是线性的。一个典型的线性规划任务如下：

$$\min\ \mathbf{c}^T\mathbf{x} \quad s.t.\ \mathbf{A}\mathbf{x} - \mathbf{b} \geq \mathbf{0}. \tag{11.23}$$

上式中$\mathbf{A}\mathbf{x} - \mathbf{b} \geq \mathbf{0}$事实上是一组限制条件，每个条件可表示为$\mathbf{a}_i^T\mathbf{x} = b_i \geq 0$，其中$\mathbf{a}_i^T$是矩阵$\mathbf{A}$的第$i$行。如图 11.8 所示，这些限制条件相当于由若干直线围成了一个合法区域，$\mathbf{c}^T\mathbf{x}$的最优值总会出现在某两条直接的交点。基于线性约束的凸函数性质，这一优化问题的局部最优解即为全局最优解，但这一最优解未必唯一。如图 11.8，如果$\mathbf{c}^T\mathbf{x} = const$的梯度$\mathbf{c}$方向发生改变时，有可能使得合法区域的整条边都是全局最优解。

上述优化任务的对偶任务有如下形式：

$$q(\boldsymbol{\lambda}) = \inf_{\mathbf{x}} \{\mathbf{c}^T\mathbf{x} - \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{x} - \mathbf{b})\} = \inf_{\mathbf{x}} (\mathbf{c}^T - \boldsymbol{\lambda}^T\mathbf{A})\mathbf{x} + \boldsymbol{\lambda}^T\mathbf{b},$$

$$\max_{\boldsymbol{\lambda}} q(\boldsymbol{\lambda}) \quad s.t.\ \boldsymbol{\lambda} \geq \mathbf{0}.$$

注意如果$\mathbf{c} - \mathbf{A}\boldsymbol{\lambda} \neq \mathbf{0}$时，$q(\boldsymbol{\lambda}) = -\infty$，这种情况显然不是对偶任务的解，因此必有$\mathbf{c} - \mathbf{A}\boldsymbol{\lambda} = \mathbf{0}$。由此可得线性规代任务的对偶问题为：

$$\max_{\boldsymbol{\lambda}} \boldsymbol{\lambda}^T\mathbf{b} \quad s.t.\ \boldsymbol{\lambda} \geq \mathbf{0},\ \mathbf{c} - \mathbf{A}\boldsymbol{\lambda} = \mathbf{0}. \tag{11.24}$$

也可以将对偶问题写成Wolfe对偶形式如下：

$$\max_{\mathbf{x},\boldsymbol{\lambda}} \mathbf{c}^T\mathbf{x} - \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) \ \ s.t.\ \mathbf{c} - \mathbf{A}\boldsymbol{\lambda} = \mathbf{0},\ \boldsymbol{\lambda} \geq \mathbf{0}.$$

将上式中的限制条件代入优化任务，则得到和式 11.24一致的形式。

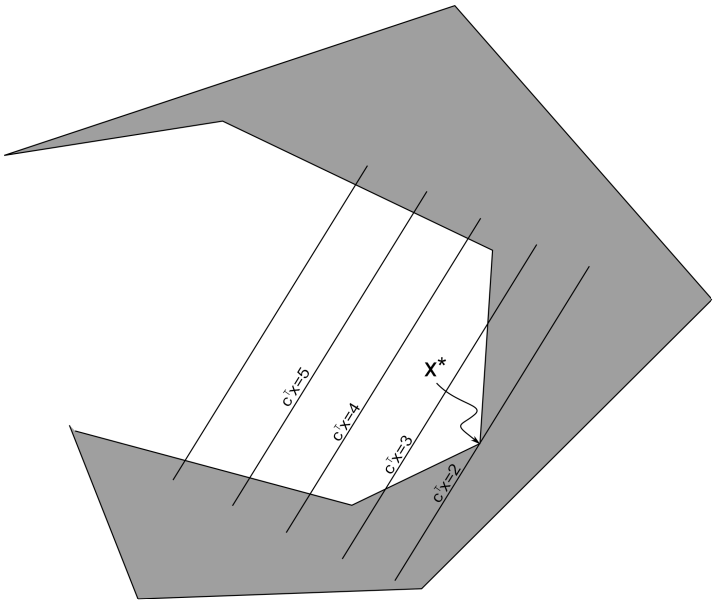

Fig. 11.8 线性规划（LP）任务。约束$\mathbf{Ax}-\mathbf{b}\geq\mathbf{0}$对应一组约束$\mathbf{a}_i^T\mathbf{x}-b_i\geq 0$，每个约束对应由一条直接分割出的半边空间。这些约束共同决定了一个合法区域，优化问题需要在这一区域中求找最优解。图中阴影部分表示非法区域，$\mathbf{x}^*$是最优解，出现在合法区域的某个顶点或某条边。

11.3.3.1 单纯型法（Simplex）

Simplex方法是解决线性优化问题最通用的方法，由Dantzig于1940年提出。前面我们讨论过，由于线性优化问题中优化目标和限制条件的凸函数属性，其合法区域是一个凸包，而优化问题的解处于该凸包的某个顶点。因此，我们可以从某一个顶点出发，依凸包的边寻找使目标函数更优化的顶点。因为凸包的顶点是有限的，因此上述搜索算法可在有限步骤内得到问题的最优解，如图 11.9所示。现在的问题是如何确定凸包中顶点的位置，以级如何依凸包的边寻找更优的顶点。

升维优化

下面我们说明凸包的顶点可以由线性方程求解得到。先看一个简单的例子。设我们有如下线性规划任务：

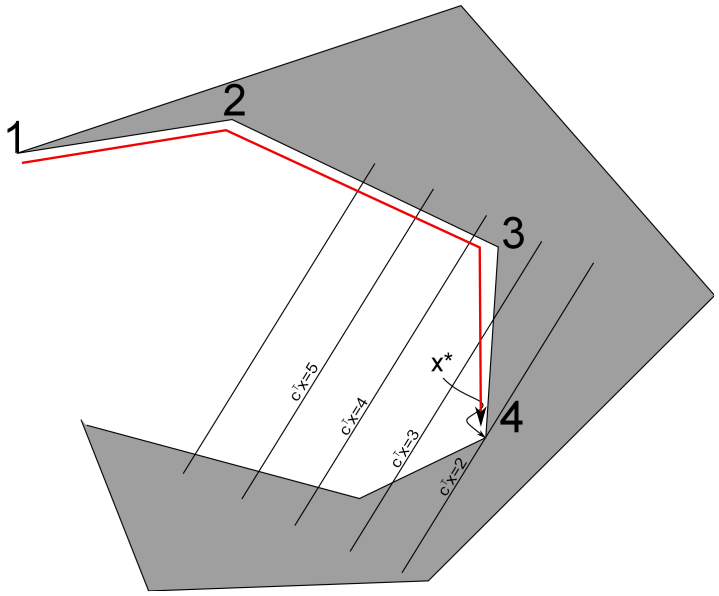

Fig. 11.9 线性规划问题的合法区域是一个凸包，Simplex算法通过搜索凸包的顶点进行优化。

$$\min_{x,y} \quad 2x-y \quad s.t.\ x+y \leq 1, x \geq 0, y \geq 0. \tag{11.25}$$

注意上式可很容易写成如式 11.23所示的形式。上述问题是一个二维空间中的带约束优化问题，其中不等式约束对应的合法区域如图 11.10（a）所示。我们知道这一问题的最优点应该在a,b,c 这三个顶点，但因为约束条件中的不等式处理起来不直观，我们可以将上述问题“升维”到三维空间，引入第三维变量z，由此可将不等约束写成等式约束，即：

$$\min_{x,y} \quad 2x-y \quad s.t. \quad x+y+z=1, x \geq 0, y \geq 0, z \geq 0. \tag{11.26}$$

容易验证，问题 11.26 与问题 11.25在对x和y的约束上是等价的，而优化函数中只包含x和y，因此这两个问题的解是一致的。图 11.10 （b）给出问题 11.26所对应的合法区域，事实上是一个三维空间中第一象限中的一个有限平面，或一个单纯型（Simiplex）。这一平面上的所有点都是问题 11.26的合法点，其相应的(x,y)座标也是问题 11.25的合法点。

由于式 11.26中的所有限制条件也是凸的，因此其合法区域也是个凸包，且最优点是这一凸包的顶点（图 11.10（b）中a,b,c三点）。通过引入变量z将问题升维到三维空间的好处是把不等关系都归结到座标轴上，从而使凸包的

顶点都置于座标轴上。这意味着只需将x,y,z中的任意两个变量置0，即可得到一个顶点，这一顶点的具体座标可由等式约束得到。在本例中，任意两个变量置零后另一个变量都为1。显然，a,b,c这三个顶点中，$a=(0,1,0)$的目标函数值最低（-1），因此是最优点。

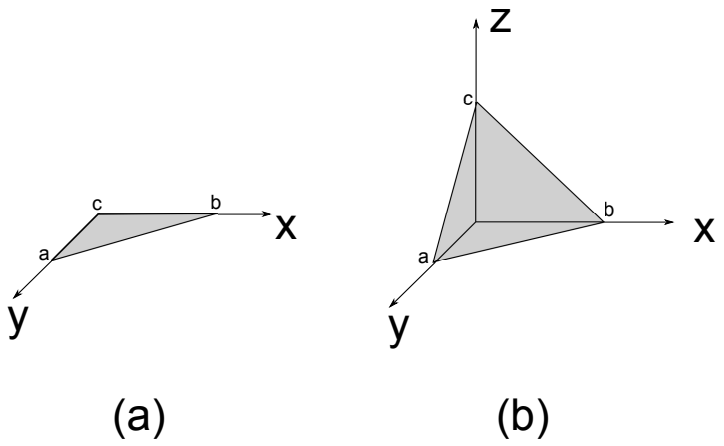

Fig. 11.10 线性规划问题的合法区域是一个凸包，Simplex算法通过搜索凸包的顶点进行优化。

上述通过引入附加变量将线性优化问题转换成求高维空间中Simplex顶点问题的方法称为单纯型法（Simplex）。我们可以将这一方法形式化如下：设有如下带有$m+n$个约束的线性优化问题：

$$\min_{\mathbf{x}} \ \mathbf{c}^T\mathbf{x} \quad s.t. \ \mathbf{A}\mathbf{x}-\mathbf{b}\geq\mathbf{0};\mathbf{x}\geq 0, \tag{11.27}$$

其中$\mathbf{x}\in R^n$, $\mathbf{b}\in R^m$，$\mathbf{A}\in R^{m\times n}$。注意这一形式和式 11.23略有不同，加入了对$\mathbf{x}$的非负约束，但式 11.23事实上可通过简单的变量替换规则写成如式 11.28的形式（[8],pp.356-357）。

式 11.28中包含m个不等约束（不包含$\mathbf{x}$上的约束），因此引入m个非负的自由变量（有时也称松驰变量）$\mathbf{x}'=x_{n+1},...,x_{n+m}$，使得优化问题 11.28的约束升维到$n+m$ 空间，得到如下约束问题：

$$\min_{\mathbf{x}} \ \mathbf{c}^T\mathbf{x} \quad s.t. \ \mathbf{A}\mathbf{x}-\mathbf{b}-\mathbf{x}'=\mathbf{0};\mathbf{x}\geq\mathbf{0};\mathbf{x}'\geq\mathbf{0}. \tag{11.28}$$

和图 11.10中类似，升维后优化问题的合法域构成的凸包中，顶点所对应的n个变量的值为零，其余m个变量非零。这m个非零变量称为基变量，n 个零变量称为非基变量。由等式约束可解出m基变量，即得到对应的顶点。

顶点移动

如果不考虑速度问题，依前述升维优化方法，不妨将$m+n$个变量中所有可能的基变量组合都列出来，得到C_{m+n}^{m}个合法区域的顶点，对比每一个顶点的目标函数值，其中最大函数值即为优化问题的解。显然，这种方法的计算复杂度随优化任务的维度呈指数增长，效率很低。我们可取更有效的策略，例如可以在得到一个顶点后，沿某一轴变量寻找下一个顶点，使得新顶点的目标函数更低。基于这一方法，通常经过几次搜索即可找到最优解。

为表达得更清晰，我们对式 11.28进行整理，将$\mathbf{x}$ 和$\mathbf{x}'$合并成一个向量，用$\tilde{\mathbf{x}}$表示，整理并重新定义相应参数，可得如下简单形式：

$$\min_{\tilde{\mathbf{x}}} \tilde{\mathbf{c}}^T\tilde{\mathbf{x}} \quad s.t.\ \ \tilde{\mathbf{A}}\tilde{\mathbf{x}} = \mathbf{b}; \tilde{\mathbf{x}} \geq \mathbf{0}. \tag{11.29}$$

其中:

$$\tilde{\mathbf{x}} = [\mathbf{x}^T \ \ \mathbf{x}'^T]^T \tag{11.30}$$

$$\tilde{\mathbf{c}} = [\mathbf{c}^T \ \ \mathbf{0}_{1\times m}]^T \tag{11.31}$$

$$\tilde{\mathbf{A}} = [\mathbf{A} \ \ \mathbf{I}_{m\times m}]. \tag{11.32}$$

在不产生歧义的前提下，我们用$\mathbf{x}$,$\mathbf{c}$,$\mathbf{A}$分别替换$\tilde{\mathbf{x}}$,$\tilde{\mathbf{c}}$,$\tilde{\mathbf{A}}$，得到优化任务的如下简单形式：

$$\min_{\mathbf{x}} \ \mathbf{c}^T\mathbf{x} \quad s.t.\ \ \mathbf{A}\mathbf{x} = \mathbf{b}; \mathbf{x} \geq \mathbf{0}. \tag{11.33}$$

这一形式称为线性优化问题的标准形式。上式中的最优解$\mathbf{x}$应满足如下KKT条件:

$$\mathbf{A}^T\boldsymbol{\lambda}+\mathbf{s}=\mathbf{c} \tag{11.34}$$

$$\mathbf{A}\mathbf{x}=\mathbf{b} \tag{11.35}$$

$$\mathbf{x}\geq\mathbf{0} \tag{11.36}$$

$$\mathbf{s}\geq\mathbf{0} \tag{11.37}$$

$$x_i s_i=0,\quad i=1,2,...,n+m \tag{11.38}$$

其中$\boldsymbol{\lambda}$为等式约束对应的拉格朗日因子，$\mathbf{s}$为不等约束对应的拉格朗日因子。记基变量集合为$\mathbf{x}_B$，其对应的索引集合为$\mathscr{B}$；记非基变量集合为$\mathbf{x}_N$，其对应的索引集合为$\mathscr{N}$。记$\mathbf{A}$ 中基变量索引对应的子矩阵为$\mathbf{B}$，非基变量索引对应的子矩阵为N。子矩阵B也称为轴矩阵，其中每一列称为一个轴（Basis）。$\mathbf{c},\mathbf{s}$中对应基变量的子向量记为$\mathbf{c}_B,\mathbf{s}_B$，对应非基变量的子向量记为$\mathbf{c}_N,\mathbf{s}_N$。注意$\mathbf{x}_N=0$，$\mathbf{x}_B>0$[1]。

由等式约束式11.35可知：

$$\mathbf{x}_B=\mathbf{B}^{-1}\mathbf{b}.$$

将式 11.34中变量分成$\mathbf{x}_B$和$\mathbf{x}_N$两个子向量，有：

$$\mathbf{B}^T\boldsymbol{\lambda}+\mathbf{s}_B=\mathbf{c}_B, \tag{11.39}$$

$$\mathbf{N}^T\boldsymbol{\lambda}+\mathbf{s}_N=\mathbf{c}_N. \tag{11.40}$$

由于$\mathbf{x}_B>0$，依式 11.38易知$\mathbf{s}_B=0$。代入公式 11.39，有：

$$\mathbf{B}^T\boldsymbol{\lambda}=\mathbf{c}_B,$$

因而有：

$$\boldsymbol{\lambda}=\mathbf{B}^{-T}\mathbf{c}_B. \tag{11.41}$$

将$\boldsymbol{\lambda}$代入式11.40，可解得$\mathbf{s}_N$如下：

$$\mathbf{s}_N=\mathbf{c}_N-\mathbf{N}^T\boldsymbol{\lambda}=\mathbf{c}_N-(\mathbf{B}^{-1}\mathbf{N})^T\mathbf{c}_B. \tag{11.42}$$

如果$\mathbf{s}_N\geq\mathbf{0}$，则说明当前的解$\mathbf{x}$满足所有KKT条件，因而是优化问题11.33的解，同时也是优化问题 11.28的解。否则，取$\mathbf{s}_N$中某一负值元素对应

[1] 基变量为零时称轴矩阵$\mathbf{B}$是退化的（Degenerate），我们不做过多讨论。读者可参考 [8]（pp.381）。

的变量x_q，可以证明当固定$\mathbf{x}_N$中除x_q之外的变量为零时，增大x_q的值总会使目标函数$\mathbf{c}^T\mathbf{x}$下降。为此，令x_q的新取值为x_q^+，对应新的$\mathbf{x}$记为$\mathbf{x}^+$（注意$\mathbf{x}^+$的第q 位元素为x_q^+）。由于$x_j = 0, j \in \mathscr{N}\backslash q$，因而有:

$$\mathbf{A}\mathbf{x}^+ = \mathbf{B}\mathbf{x}_B^+ + \mathbf{a}_q x_q^+,$$

其中$\mathbf{a}_q$是$\mathbf{A}$中的第q列。因为$\mathbf{B}\mathbf{x}_B = \mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{x}^+$，有：

$$\mathbf{x}_B^+ = \mathbf{x}_B - \mathbf{B}^{-1}\mathbf{a}_q x_q^+. \tag{11.43}$$

求$\mathbf{x}^+$处的目标函数，有：

$$\mathbf{c}^T\mathbf{x}^+ = \mathbf{c}_B^T\mathbf{x}_B^+ + c_q x_q^+ = \mathbf{c}_B^T\mathbf{x}_B - \mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{a}_q x_q^+ + c_q x_q^+ \tag{11.44}$$

由式 11.41 知:

$$\mathbf{c}_B^T\mathbf{B}^{-1} = \boldsymbol{\lambda}^T;$$

由 11.42知：

$$\mathbf{a}_q^T\boldsymbol{\lambda} = c_q - s_q;\ \ q \in N.$$

将上面二式代入式 11.44可得：

$$\begin{aligned}
\mathbf{c}^T\mathbf{x}^+ &= \mathbf{c}_B^T\mathbf{x}_B - \mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{a}_q x_q^+ + c_q x_q^+ && (11.45)\\
&= \mathbf{c}_B^T\mathbf{x}_B - \boldsymbol{\lambda}^T\mathbf{a}_q x_q^+ + c_q x_q^+ && (11.46)\\
&= \mathbf{c}_B^T\mathbf{x}_B - (c_q - s_q)x_q^+ + c_q x_q^+ && (11.47)\\
&= \mathbf{c}^T\mathbf{x} + s_q x_q^+. && (11.48)
\end{aligned}$$

如果我们选择$s_q < 0$，则总有$\mathbf{c}^T\mathbf{x}^+ < \mathbf{c}^T\mathbf{x}$，因此$\mathbf{x}^+$总比$\mathbf{x}$更优。事实上我们可以一直增大$x_q^+$，直到$\mathbf{x}_B$中某一个元素$x_p$变成0，这事实上相当于沿着合法域凸包的一条边走从一个顶点走到了另一个顶点。到达该顶点后，基变量索引集与非基变量索引集分别变成：

$$\mathscr{B}^+ = \mathscr{B}\backslash p \cup \{q\},$$
$$\mathscr{N}^+ = \mathscr{N}\backslash q \cup \{p\}.$$

上述变动可以理解为在矩阵$\mathbf{A}$中选择基矩阵$\mathbf{B}$时，去掉第p列而选择第q列，其中选择q使得$s_q < 0$，而p是当x_q增大时，轴$\mathbf{x}_B$中最先被先被降

为0的x_p所对应的p。$\mathbf{x}_B$依x_q的变化可由式11.43得到，即:

$$\mathbf{x}_B^+ = \mathbf{x}_B - \mathbf{B}^{-1}\mathbf{a}_q x_q^+.$$

首先计算$\mathbf{d} = \mathbf{B}^{-1}\mathbf{a}_q$，这可由解方程组得$\mathbf{Bd} = \mathbf{a}_q$得到；再对$\mathscr{D}$中的每个$k$，计算使得$(\mathbf{x}_B^+)_k = (\mathbf{x}_B - \mathbf{d}x_q^+)_k = 0$ 对应的x_q^+值，并取x_q^+最小的k作为需要去掉的基，即：

$$p = \arg\min_k \frac{(\mathbf{x}_B)_k}{d_k}.$$

上述算法从基矩阵$\mathbf{B}$中去掉轴$\mathbf{a}_p$，并加入新的轴$\mathbf{a}_q$。这一操作也称为‘转轴操作’[2]。Simplex方法每次迭代做一次转轴操作，每次操作保证新的顶点更优。在绝大部分应用里，Simplex效率非常高，一般在几次更新后即可找到全局最优点，但在一些特殊例子里仍需要遍历所有顶点，因此这一算法在最差情况下的复杂度依然是指数级的。

11.3.3.2 内点法（Interior-point）

单纯形法沿着合法域的边寻找对应最优解的顶点，这一方法在一些问题上效率较低。内点法（Interior-point）是另一种常用的线性规划算法。和单纯形法不同的是，内点法从合法域内的某一点开始迭代寻找，在保证搜索点在合法域内的同时，使目标函数值越来越小，如图 11.11所示。

Primal-Dual算法是一种常见的内点法。直观上，线性规化的原问题和对偶问题都包含多个限制条件，形式复杂。在搜索过程中保证这些复杂的限制条件得以满足是非常困难的。Primal-Dual算法将优化问题转化成对KKT条件的求解问题，通过求满足KKT条件的解来间接实现对原问题的优化。

设要优化的线性规划问题有式 11.33所示的标准形：

$$\min_{\mathbf{x}} \ \mathbf{c}^T\mathbf{x} \quad s.t. \ \mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \tag{11.49}$$

其对应的KKT条件如式 11.34-11.38，重写如下：

[2] https://zh.wikipedia.org/wiki/单纯形法

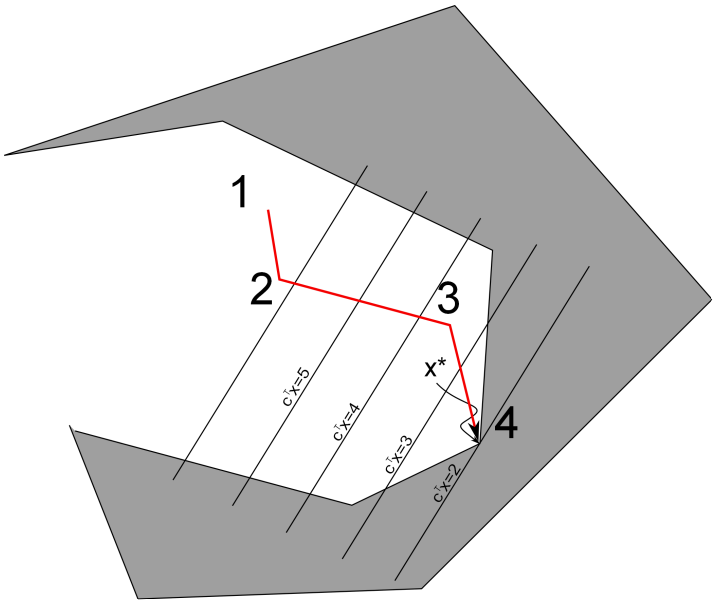

Fig. 11.11 内点法。由初始点开始，在合法域内搜索，使得目标函数的取值逐渐降低。图中红线表示搜索路径。

$$\mathbf{A}^T\boldsymbol{\lambda}+\mathbf{s}=\mathbf{c} \tag{11.50}$$

$$\mathbf{A}\mathbf{x}=\mathbf{b} \tag{11.51}$$

$$\mathbf{x}\geq\mathbf{0} \tag{11.52}$$

$$\mathbf{s}\geq\mathbf{0} \tag{11.53}$$

$$x_i s_i=0;\quad i=1,2,...,n+m \tag{11.54}$$

注意上述问题（式 11.49）的对偶问题是:

$$\max_{\boldsymbol{\lambda}}\mathbf{b}^T\boldsymbol{\lambda}\quad s.t.\ \mathbf{A}^T\boldsymbol{\lambda}+\mathbf{s}=\mathbf{c},\mathbf{s}\geq 0 \tag{11.55}$$

其KKT条件和原问题是一样的。这意味着我们只要找到一组满足上述KKT条件的解$(\mathbf{x}^*,\boldsymbol{\lambda}^*,\mathbf{s}^*)$，它同时也是原问题 11.49和其对偶问题 11.55的解。

对KKT条件11.50 - 11.66 进行整理，将式 11.50,11.51,11.66 合并成如下形式：

$$F(\mathbf{x},\boldsymbol{\lambda},\mathbf{s}) = \begin{bmatrix} \mathbf{A}^T\boldsymbol{\lambda} + \mathbf{s} - \mathbf{c} \\ \mathbf{A}\mathbf{x} - \mathbf{b} \\ \mathbf{XSe} \end{bmatrix} = 0, \tag{11.56}$$

其中: $\mathbf{X} = diag(x_1,...,x_n), \mathbf{S} = diag(s_1,...,s_n), \mathbf{e} = (1,1,...,1)^T$。注意KKT条件是原问题 11.49 （或对偶问题 11.55）的充分条件，因此原带约束优化问题转化为如下带约束求解问题：

$$F(\mathbf{x},\boldsymbol{\lambda},\mathbf{s}) = 0 \quad s.t. \ \mathbf{x} \geq \mathbf{0}, \mathbf{s} \geq \mathbf{0}. \tag{11.57}$$

假设$F(\mathbf{x},\boldsymbol{\lambda},\mathbf{s}) = 0$很难求解（否则立即可得到该解并判断其是否符合约束条件），因此需要用迭代法逐渐逼近。牛顿法是常用的方法。这一方法可用图 11.12解释。设有一元函数$f(x)$，我们欲求使$f(x) = 0$的解x^*。牛顿法从某一初值x_0开始搜索，设第k 次迭代时的值是$\mathbf{x}_k$，则取该点处的切线与x轴的交点作为新的取值x_{k+1}。注意该切线的斜率为$f(x)$在x_k点处的导数$f'(x_k)$，因而有：

$$\frac{f(x_k)}{x_{k+1} - x_k} = -f'(x_k).$$

注意公式中的负号表示当导数为正数时，$x_{k+1} < x_k$。记$\Delta x|k = x_{k+1} - x_k$，上式可写成:

$$f'(x_k)\Delta x|k = -f(x).$$

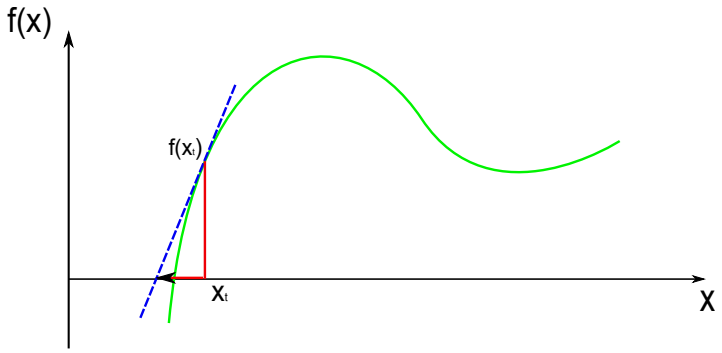

Fig. 11.12 一元函数的牛顿法示例。每次迭代取$f(x)$在当前点的切线与x轴的交点作为新的优化点。

扩展到多元函数情况，牛顿法的更新形式如下：

$$\nabla F(\mathbf{x}_k)\Delta\mathbf{x}|k = -F(\mathbf{x}_k),$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}|k.$$

将上式应用于式 11.57，可得：

$$\nabla F(\mathbf{x}_k, \boldsymbol{\lambda}_k, \mathbf{s}_k) \begin{bmatrix} \Delta\mathbf{x}|k \\ \Delta\boldsymbol{\lambda}|k \\ \Delta\mathbf{s}|k \end{bmatrix} = -F(\mathbf{x}_k, \boldsymbol{\lambda}_k, \mathbf{s}_k).$$

代入$F(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s})$的形式11.56，有：

$$\begin{bmatrix} \mathbf{0} & \mathbf{A}^T & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S} & \mathbf{0} & \mathbf{X} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x}|k \\ \Delta\boldsymbol{\lambda}|k \\ \Delta\mathbf{s}|k \end{bmatrix} = \begin{bmatrix} -(\mathbf{A}^T\boldsymbol{\lambda} + \mathbf{s} - \mathbf{c}) \\ -(\mathbf{A}\mathbf{x} - \mathbf{b}) \\ -(\mathbf{X}\mathbf{S}\mathbf{e}) \end{bmatrix}.$$

对上式进行求解，得到$[\Delta\mathbf{x}|k\ \Delta\boldsymbol{\lambda}|k\ \Delta\mathbf{s}|k]$之后，即可得到$k+1$时刻的变量：

$$[x_{k+1}\ \boldsymbol{\lambda}_{k+1}\ \mathbf{s}_{k+1}] = [\mathbf{x}_k\ \boldsymbol{\lambda}_k\ \mathbf{s}_k] + [\Delta\mathbf{x}|k\ \Delta\boldsymbol{\lambda}|k\ \Delta\mathbf{s}|k].$$

但上式有可能超出合法域，即违反$\mathbf{x} \geq \mathbf{0}, \mathbf{s} \geq \mathbf{0}$的约束，因此需限制更新的幅度。这可以通过加入一个$(0,1)$之间的步长因子$\alpha$ 实现，即：

$$[\mathbf{x}_{k+1}\ \boldsymbol{\lambda}_{k+1}\ \mathbf{s}_{k+1}] = [\mathbf{x}_k\ \boldsymbol{\lambda}_k\ s_k] + \alpha[\Delta\mathbf{x}|k\ \Delta\boldsymbol{\lambda}|k\ \Delta\mathbf{s}|k].$$

上述过程是Primal-Dual算法的基本原理。在实际实现时，有多种方法控制每一步的迭代步长，例如考虑当前点$(\mathbf{x}_k, \boldsymbol{\lambda}_k, \mathbf{s}_k)$ 对非负约束$\mathbf{x} \geq \mathbf{0}, \mathbf{s} \geq \mathbf{0}$的‘违反程度’，可用$\mu_k = \mathbf{x}_k^T\mathbf{s}_k$。显然$\mu_k$越小，表示当前取值点越接近合法区域的边界，因而需要减小步长。具体算法可参考 [8]（pp.417）。

11.3.4 二阶规划

二阶规划（Quadratic Programming, QP）是优化目标为二阶函数，约束为线性函数的优化任务，定义如下：

$$\begin{aligned} &\min_{\mathbf{x}} q(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T\mathbf{G}\mathbf{x} + \mathbf{x}^T\mathbf{c} \\ &s.t.\ \ \mathbf{a}_i^T\mathbf{x} = b_i,\ \ i \in \mathscr{E};\ \ \ \mathbf{a}_i^T\mathbf{x} \geq b_i\ i \in \mathscr{I}. \end{aligned} \tag{11.58}$$

其中$\mathscr{E}$是等式约束集，$\mathscr{I}$是不等约束集。和线性规划相比，二阶规划多出一个二阶项$\mathbf{x}^T\mathbf{G}\mathbf{x}$。我们会看到，二阶规划任务的难度和二阶矩阵$\mathbf{G}$直接相关，

当**G**为正定矩阵时，其复杂性和线性规划相当，否则优化任务要复杂的多。二阶规划是非线性优化问题的特例，不仅本身有重要应用价值，对解决一般非线性优化问题也具有重要意义：我们会看到一般非线性问题可以通过构造局部QP问题实现。

11.3.4.1 等式约束

我们首先考虑比较简单的等式约束的情况，即:

$$\min_{\mathbf{x}} q(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{G}\mathbf{x} + \mathbf{x}^T\mathbf{c} \quad s.t.\ \mathbf{a}_i^T\mathbf{x} = b_i,\ \ i \in \mathscr{E}. \tag{11.59}$$

上述问题的拉格朗日目标函数为：

$$L(\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{2}\mathbf{x}^T\mathbf{G}\mathbf{x} + \mathbf{x}^T\mathbf{c} - \boldsymbol{\lambda}(\mathbf{A}\mathbf{x} - \mathbf{b}).$$

该问题的解应满足如下条件：

$$\frac{\partial L}{\partial \mathbf{x}} = \mathbf{G}\mathbf{x} + \mathbf{c} - \boldsymbol{\lambda}\mathbf{A} = \mathbf{0}$$

$$\frac{\partial L}{\partial \boldsymbol{\lambda}} = \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0}.$$

写成矩阵形式，有：

$$\begin{bmatrix} \mathbf{G} & -\mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \lambda \end{bmatrix} = \begin{bmatrix} -\mathbf{c} \\ \mathbf{b} \end{bmatrix}. \tag{11.60}$$

可以证明，如果**A**是行满秩的，且**G**是正定的，则满足上式的**x**和**y**即是原问题 11.59的全局唯一最优解。如果**G**是半正定的（即其特征值中包含零值），则该解是局部最优解；如果**G** 是非定的（特征值中既包含正值，也包含负值），则该解只是一个驻点，并非局部最优（例如可能是一个马鞍点）。

11.3.4.2 不等约束

下面我们讨论引入不等约束的情形。依拉格朗日乘子法，QP问题11.58的拉格朗日目标函数为：

$$\min \frac{1}{2}\mathbf{x}^T\mathbf{G}\mathbf{x} + \mathbf{x}^T\mathbf{c} - \sum_{i \in \mathscr{E} \cup \mathscr{I}} \lambda_i(\mathbf{a}_i^T\mathbf{x} - b_i). \tag{11.61}$$

对上式的优化解($\mathbf{x}^*$,$\boldsymbol{\lambda}^*$应满足KKT条件，即：

$$\mathbf{G}\mathbf{x}^* + \mathbf{c} - \sum_i \lambda_i^* \mathbf{a}_i = 0 \tag{11.62}$$

$$\mathbf{a}_i^T \mathbf{x}^* = b_i, \ \ i \in \mathscr{E} \tag{11.63}$$

$$\mathbf{a}_i^T \mathbf{x}* \geq b_i, \ \ i \in \mathscr{I} \tag{11.64}$$

$$\lambda_i^* \geq 0, \ \ i \in \mathscr{I} \tag{11.65}$$

$$\lambda_i^* (\mathbf{a}_i^T \mathbf{x}^* - b_i) = 0, \ \ i \in \mathscr{I}. \tag{11.66}$$

如果我们考察$\mathbf{x}^*$点在不等约束上的满足情况，会发现该点在某些约束的边界上，因而这些约束退化成等式约束。这些以等式形式出现的约束称为*激活约束*，激活约束组成的集合称为*激活集*（Active Set），记为$\mathscr{A}(\mathbf{x}^*)$，即：

$$\mathscr{A}(\mathbf{x}^*) = \{i \in \mathscr{E} \cup \mathscr{I} | \mathbf{a}_i^T \mathbf{x}^* = b_i\}.$$

因而前述的KKT约束可重写如下：

$$\mathbf{G}\mathbf{x}^* + \mathbf{c} - \sum_i \lambda_i^* \mathbf{a}_i = \mathbf{0} \tag{11.67}$$

$$\mathbf{a}_i^T \mathbf{x}^* = b_i, \ \ i \in \mathscr{A}(x^*) \tag{11.68}$$

$$\mathbf{a}_i^T \mathbf{x}^* > b_i, \ \ i \in I \backslash \mathscr{A}(x^*) \tag{11.69}$$

$$\lambda_i^* \geq 0, \ \ i \in \mathscr{I} \cap \mathscr{A}(x^*) \tag{11.70}$$

可以证明，如果$\mathbf{G}$是正定的，则满足式 11.67-11.70 的$\mathbf{x}^*$是二阶规划问题11.59的全局最优解。如果$\mathbf{G}$不是正定的，则一般不存在全局最优解。我们仅考虑G是正定的情况；更复杂的情况请参考本章结尾的‘相关资源’一节。

激活集算法

我们介绍激活集方法。直观上，如果我们能确定一个最优激活集，则可将不等约束的QP问题转化成等式约束的QP问题，即：

$$\min \frac{1}{2}\mathbf{x}^T \mathbf{G}\mathbf{x} + \mathbf{x}^T \mathbf{c} \quad s.t. \quad \mathbf{a}_i^T \mathbf{x} = b_i, i \in \mathscr{A}(x^*).$$

问题的复杂之处在于我们并不能事先知道这一最优激活集，因此需要进行搜索。最简单的办法是逐一列举所有可能的激活集，考察在某一激活集上是否存在满足所有KKT条件的解$(\mathbf{x}^*, \boldsymbol{\lambda}^*)$，如果存在，则该解必为全局最优解。这种盲目搜索需要考察2^m个可能的激活集，其中m为约束个数，显然效率太低。

激活集算法基于一个迭代过程逐渐发现合理的基活集，并由此逐渐找到最优解。设第k次迭代时的解为$\mathbf{x}_k$，激活集为$\mathscr{W}_k$。我们将保持$\mathscr{W}_k$ 中的等式约束对$q(\mathbf{x})$进行最小化。记新解的位移为$\mathbf{p} = \mathbf{x} - \mathbf{x}_k$, 并记$\mathbf{g}_k = \mathbf{G}\mathbf{x}_k + \mathbf{c}$, 则第$k$步的优化目标可写成：

$$q(\mathbf{x}) = q(\mathbf{x}_k + \mathbf{p}) = \frac{1}{2}\mathbf{x}^T\mathbf{G}\mathbf{x} + \mathbf{c}^T\mathbf{x} = \frac{1}{2}\mathbf{p}^T\mathbf{G}\mathbf{p} + \mathbf{g}_k^T\mathbf{p} + \rho_k, \tag{11.71}$$

其中ρ是与$\mathbf{p}$无关的量:

$$\rho_k = \frac{1}{2}\mathbf{x}_k^T\mathbf{G}\mathbf{x}_k + \mathbf{c}^T\mathbf{x}_k.$$

因此对$q(\mathbf{x})$的优化问题转化为:

$$\min_{\mathbf{p}} \frac{1}{2}\mathbf{p}^T\mathbf{G}\mathbf{p} + \mathbf{g}_k^T\mathbf{p} \quad s.t. \quad \mathbf{a}_i^T\mathbf{p} = 0 \;\; i \in \mathscr{W}_k. \tag{11.72}$$

上式是一个仅有等式约束的QP问题，可由前一节介绍的包含等式约束的QP问题的求解方法进行优化。设上述优化得到的解为$\mathbf{p}_k$，则$\mathbf{x}_k + \mathbf{p}_k$显然满足$\mathscr{W}_k$中的所有等式约束，但未必满足$\mathscr{W}_k$中的不等约束。为此，可以给定一个$[0,1]$间的尺度因子$\alpha_k$对$\mathbf{p}_k$进行步长调整，即：

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k. \tag{11.73}$$

为使$\mathbf{x}_{k+1}$满足不等约束，要求：

$$\mathbf{a}_i^T\mathbf{x}_{k+1} = \mathbf{a}_i^T\mathbf{x}_k + \alpha_i\mathbf{a}_i^T\mathbf{p}_k \geq \mathbf{a}_i^T\mathbf{x}_k \geq b_i \quad i \notin \mathscr{W}_k$$

整理可得：

$$\alpha_k \leq \frac{b_i - \mathbf{a}_i^T\mathbf{x}_k}{\mathbf{a}_i^T\mathbf{p}_k}.$$

选择使所有不等约束满足的最大值作为α_k，有：

$$\alpha_k = \min(1, \min_{i \notin \mathscr{W}} \frac{b_i - \mathbf{a}_i^T\mathbf{x}_k}{\mathbf{a}_i^T\mathbf{p}_k}). \tag{11.74}$$

注意如果$\alpha_k < 1$，则说明$\mathbf{x}_k$的更新受到了某一个或多个不在$\mathcal{W}_k$中的不等约束的限制。基于α_k的计算方法，可知$\mathbf{x}_{k+1}$必然处在某一不等约束的合法区域边界，该约束对应的$\frac{b_i - \mathbf{a}_i^T \mathbf{x}_k}{\mathbf{a}_i^T \mathbf{p}_k}$取值最小。将这一约束加入$\mathcal{W}_k$中作为新的激活约束，因此有：

$$\mathcal{W}_{k+1} = \mathcal{W}_k \cup \{j\}, \quad j = \underset{i, \mathbf{a}_i^T \mathbf{p}_k < 0}{\arg\min} \frac{b_i - \mathbf{a}_i^T \mathbf{x}_k}{\mathbf{a}_i^T \mathbf{p}_k}. \tag{11.75}$$

上述过程重复进行，直到$\mathbf{p}_k = 0$。记此时的激活集为$\hat{\mathcal{W}}$，解为$\hat{\mathbf{x}}$，对应的$\hat{\boldsymbol{\lambda}}_i$可由KKT条件 11.67 得到：

$$\sum_{i \in \hat{\mathcal{W}}} \hat{\lambda}_i \mathbf{a}_i = \mathbf{G}\hat{\mathbf{x}} + \mathbf{c}. \tag{11.76}$$

注意上述$\hat{\boldsymbol{\lambda}}$值是以$\hat{\mathcal{W}}$作为激活集对式 11.72进行优化的结果，对$\hat{\boldsymbol{\lambda}}$ 的取值没有限制。然而，对原优化问题11.61，如果在不等约束集$\mathcal{I}$中的约束i在$\hat{\mathbf{x}}$点被激发，则应有$\hat{\lambda}_i \geq 0$，否则这一约束不该被激发，因为在该约束边界内的点$\mathbf{x}$可能比在约束边界上的点$\hat{\mathbf{x}}$有更好目标函数值。因此，我们需要将$\hat{\lambda}_i < 0$对应的约束从$\hat{\mathcal{W}}$中去除，再基于新的$\mathcal{W}$重复前面的优化过程。上述对激发集进行增减的过程和上一节论过的Simplex方法中的转轴方法操作是类似的。事实上Simplex算法是激活集算法的一个特例，只不过当时的目标函数是线性的，且将所有不等关系集中到了单变量上。算法 2给出了激活集算法的基本流程。

Initialization: $\mathbf{x}_0, \mathscr{W}_0 = \mathscr{A}(\mathbf{x}_0)$;
for *k:=1,2,...* **do**
  Solve problem 11.72 to get $\mathbf{p}_k$;
  //get a reasonable solution assuming equal constraints
  if $\mathbf{p}_k = 0$ **then**
    $\hat{\mathscr{W}} = \mathscr{W}_k$;
    $\hat{\mathbf{x}} = \mathbf{x}_k$;
    Compute $\hat{\lambda}_i$ by Equation 11.76;
    if $\hat{\lambda}_i \geq 0 \quad \forall i \in \mathscr{W}_k \cap \mathscr{I}$ **then**
      $\mathbf{x}^* = \hat{\mathbf{x}}$;
      break;
    end
    else
      $j = \arg\min_{j \in \mathscr{W}_k \cup \mathscr{I}} \hat{\lambda}_j$;
      $\mathscr{W}_{k+1} = \mathscr{W}_k \backslash \{j\}$;
    end
  end
  //adjust equal constrains
  else
    Compute α_k according to Equation 11.74;
    $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$;
    if $\alpha_k < 1$ **then**
      //add new constraint into the active set according to 11.75
      $\mathscr{W}_{k+1} = \mathscr{W}_k \cup \{j\}, \quad j = \arg\min_{i, \mathbf{a}_i^T \mathbf{p}_k < 0} \frac{b_i - \mathbf{a}_i^T \mathbf{x}_k}{\mathbf{a}_i^T \mathbf{p}_k}$;
    end
    else
      $\mathscr{W}_{k+1} = \mathscr{W}_k$
    end
  end
end

Algorithm 2: 激发集算法。

内点算法

在线性规划问题中我们讨论过内点法。内点法通过在合法域内做局部优化并进行小尺度更新，同时保证每次更新满足约束条件。这一方法很容易扩展到QP问题中。为简便起见，仅考虑带不等约束的优化问题如下：

$$\min \frac{1}{2}\mathbf{x}^T\mathbf{G}\mathbf{x}+\mathbf{x}^T\mathbf{c} \quad s.t.\ \mathbf{A}\mathbf{x}\geq\mathbf{b}, \tag{11.77}$$

其KKT条件为：

$$\mathbf{G}\mathbf{x}+\mathbf{c}-\mathbf{A}^T\boldsymbol{\lambda}=\mathbf{0} \tag{11.78}$$

$$\mathbf{A}\mathbf{x}-\mathbf{b}\geq\mathbf{0} \tag{11.79}$$

$$(\mathbf{A}\mathbf{x}-\mathbf{b})_i\lambda_i=0,\ \ i=1,2,...m \tag{11.80}$$

$$\boldsymbol{\lambda}\geq\mathbf{0}. \tag{11.81}$$

上式中条件 11.79比较复杂，可引入一个松驰变量$\boldsymbol{\xi}\geq 0$将该式变成一个等式约束和一个不等约束：

$$\mathbf{G}\mathbf{x}+\mathbf{c}-\mathbf{A}^T\boldsymbol{\lambda}=\mathbf{0} \tag{11.82}$$

$$\mathbf{A}\mathbf{x}-\mathbf{b}-\boldsymbol{\xi}=0 \tag{11.83}$$

$$\xi_i\lambda_i=0,\quad i=1,2,...m \tag{11.84}$$

$$(\boldsymbol{\xi},\boldsymbol{\lambda})\geq\mathbf{0} \tag{11.85}$$

如果$\mathbf{G}$是正定的，满足上述KKT条件的解$(\mathbf{x}^*,\boldsymbol{\mu}^*,\boldsymbol{\lambda}^*)$必是问题 11.86的唯一全局最优解。将上述条件写成如下简单形式：

$$F(\mathbf{x},\boldsymbol{\xi},\boldsymbol{\lambda})=\begin{bmatrix}\mathbf{G}\mathbf{x}-\mathbf{A}^T\boldsymbol{\xi}+\mathbf{c}\\ \mathbf{A}\mathbf{x}-\boldsymbol{\xi}-\mathbf{b}\\ \boldsymbol{\Xi}\boldsymbol{\Lambda}\mathbf{e}\end{bmatrix}=\mathbf{0}\ \ s.t.\ \ (\boldsymbol{\xi},\boldsymbol{\lambda})\geq\mathbf{0}, \tag{11.86}$$

其中$\boldsymbol{\Xi}=diag(\xi_1,\xi_2,...,\xi_m),\boldsymbol{\Lambda}=diag(\lambda_1,\lambda_2,...,\lambda_m),\mathbf{e}=[1,1,...,1]^T$。

内点法通过迭代求解式 11.86中的$(\mathbf{x},\boldsymbol{\xi},\boldsymbol{\lambda})$并保证每一步求解满足$(\boldsymbol{\xi},\boldsymbol{\lambda})\geq\mathbf{0}$。和线性规划中类似，我们可用牛顿法对上式进行求解。设第k次迭代的变量取值为$(\mathbf{x}_k,\boldsymbol{\xi}_k,\boldsymbol{\lambda}_k)$，依牛顿法有如下关系：

$$\nabla F(\mathbf{x}_k,\boldsymbol{\xi}_k,\boldsymbol{\lambda}_k)\begin{bmatrix}\Delta\mathbf{x}|k\\ \Delta\boldsymbol{\xi}|k\\ \Delta\boldsymbol{\lambda}|k\end{bmatrix}=-F(\mathbf{x}_k,\boldsymbol{\xi}_k,\boldsymbol{\lambda}_k).$$

代入$F(\mathbf{x},\boldsymbol{\xi},\boldsymbol{\lambda})$的具体形式，有：

$$\begin{bmatrix}\mathbf{G} & \mathbf{0} & -\mathbf{A}^T\\ \mathbf{A} & -\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\lambda} & \boldsymbol{\Xi}\end{bmatrix}\begin{bmatrix}\Delta\mathbf{x}|k\\ \Delta\boldsymbol{\xi}|k\\ \Delta\boldsymbol{\lambda}|k\end{bmatrix}=\begin{bmatrix}-(\mathbf{G}\mathbf{x}-\mathbf{A}^T\boldsymbol{\lambda}+\mathbf{c})\\ -(\mathbf{A}\mathbf{x}-\boldsymbol{\xi}-\mathbf{b})\\ -(\boldsymbol{\Lambda}\boldsymbol{\Xi}\mathbf{e})\end{bmatrix}.$$

解上述方程组得到$(\Delta\mathbf{x}|k,\Delta\boldsymbol{\xi}|k,\Delta\boldsymbol{\lambda}|k)$，即可计算$k+1$时刻的变量值如下：

$$[\mathbf{x}_{k+1}\ \boldsymbol{\xi}_{k+1}\ \boldsymbol{\lambda}_{k+1}=[\mathbf{x}_k\ \boldsymbol{\xi}_k\ \boldsymbol{\lambda}_k]+\alpha_k[\Delta\mathbf{x}|k\ \Delta\boldsymbol{\xi}|k\ \Delta\boldsymbol{\lambda}|k],$$

其中α_k是为满足非负约束$(\boldsymbol{\xi}_{k+1},\boldsymbol{\lambda}_{k+1})\geq\mathbf{0}$所取的步长。

11.3.5 一般非线性优化

对一般非线性带约束优化问题，一般先将约束问题转化为无约束问题，再用无约束问题的求解方法求局部最优。另一种方法是将一般非线性带约束优化问题分解成一系列局部优化问题，每一个局部优化问题是一个相对简单的LP问题或QP问题。

11.3.5.1 惩罚法

将约束问题转化成无约束问题的一种方法是是将绝对约束转化成‘软约束’，将约束项作为惩罚加入到目标函数中，从而将约束任务转化成无约束任务。当惩罚项在无约束任务的目标函数中比例越来越高时，软约束逐渐加强，趋向原约束任务。

设优化任务如下：

$$\min f(\mathbf{x})\quad s.t.\ c_i(\mathbf{x})=0,i\in\mathscr{E};\ \ c_i(\mathbf{x})\geq 0,i\in\mathscr{I}\tag{11.87}$$

对应的拉格朗日目标函数为：

$$L(\mathbf{x},\boldsymbol{\lambda})=f(\mathbf{x})-\sum_{i\in\mathscr{E}}\lambda_i c_i(x)-\sum_{i\in\mathscr{I}}\lambda_i c_i(\mathbf{x})\quad s.t.\ \ \lambda_i\geq 0\ \ i\in\mathscr{I}.\tag{11.88}$$

将任务中的约束条件作为惩罚项加入到目标函数$f(\mathbf{x})$中，从而将约束任务转化成无约束任务：

$$\min f(\mathbf{x}) + \frac{\mu}{2}\sum_{i\in\mathscr{E}}(c_i(\mathbf{x}))^2 + \frac{\mu}{2}\sum_{i\in\mathscr{I}}([c_i(\mathbf{x})]^-)^2 \tag{11.89}$$

其中μ是惩罚系数，$(c_i(\mathbf{x}))^2$表示等式约束的违反度，$[c_i(\mathbf{x})]^- = \max(0, -c_i(\mathbf{x}))$表示非负约束的违反度。

显而易见，μ越大对违反约束的惩罚越大，无约束任务11.89与原约束任务11.87越接近；当$\mu \to \infty$时，二者趋向相同解。将约束任务转化为无约束任务，可充分利用无约束任务上已知的各种优化方法，极大方便了对约束任务的求解。在实际实现时，通常先取一个较小的μ，基于该值对无约束问题11.89 求近似解，以此近似解作为起点，增大μ值求解新一轮无约束问题。如此迭代进行，直到收敛。

式 11.89中的惩罚项是二阶规范，因此上述方法也称为二阶惩罚方法（Quadratic Penalty Method）。另一种常用的惩罚是一阶规范，即：

$$\min f(\mathbf{x}) + \mu\sum_{i\in\mathscr{E}}|c_i(\mathbf{x})| + \mu\sum_{i\in\mathscr{I}}[c_i(\mathbf{x})]^-, \tag{11.90}$$

其中μ是惩罚系数。可以证明，当μ足够大时（具体说，大于最大的拉格朗日乘子），原带约束问题 11.87的解也是问题 11.90的解（证明可见 [8], Theorem 17.3）。

11.3.5.2 增广拉格朗日方法：等式约束

当μ取有限值时，通过对二阶惩罚的目标函数进行优化得到的$\mathbf{x}$显然并不完全满足限制性条件。首先考虑仅包含等式约束的情况，其优化任务为：

$$\min f(\mathbf{x}) \quad s.t.\ c_i(\mathbf{x}) = 0,\ \ i \in \mathscr{E}$$

其拉格朗日目标函数为：

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \sum_{i\in\mathscr{E}}\lambda_i c_i(\mathbf{x}).$$

对该任务构造带二阶惩罚的目标函数：

$$Q(\mathbf{x};\mu) = f(\mathbf{x}) + \frac{\mu}{2}\sum_{i\in\mathscr{E}}||c_i(\mathbf{x})||^2,$$

其中μ是惩罚参数。对上式进行优化得到$\tilde{\mathbf{x}}$，可以证明$c_i(\tilde{\mathbf{x}})$系统地偏离零点：

$$c_i(\tilde{\mathbf{x}}) \approx -\lambda_i^*/\mu \quad \forall i \in \mathscr{E}.$$

这说明对$Q(\mathbf{x};\mu)$不论如何优化都无法满足约束条件。如果可以设计一种带惩罚的目标函数，其优化点$\tilde{\mathbf{x}}$对约束的违反程度更小，则可基于较弱的惩罚（μ较小）得到较好的结果。

增广拉格朗日方法（Augmented Lagrangian Method）即是这种方法。通过对格拉格朗日目标函数$L(\mathbf{x},\boldsymbol{\lambda})$的近似函数进行二阶惩罚,可使得到的$\tilde{\mathbf{x}}$更符合约束条件。增广拉格朗日方法的目标函数形式如下：

$$L_A(\mathbf{x};\boldsymbol{\lambda},\mu) = f(\mathbf{x}) - \sum_{i\in\mathscr{E}} \lambda_i c_i(\mathbf{x}) + \frac{\mu}{2}\sum_{i\in\mathscr{E}} ||c_i(\mathbf{x})||^2. \tag{11.91}$$

和二阶惩罚的目标函数$Q(\mathbf{x};\mu)$相比，$L_A(\mathbf{x};\boldsymbol{\lambda},\mu)$中加入了一个约束项$\sum_{i\in\mathscr{E}}\lambda_i c_i(\mathbf{x})$。加入这一项后目标函数的形式等同于在格拉格朗日目标$L(\mathbf{x},\boldsymbol{\lambda})$上加入了一个二阶惩罚。特别注意的是，这里的$\boldsymbol{\lambda}$是一个参数，并不是一个和$\mathbf{x}$同时优化的量，因此$L_A(\mathbf{x};\boldsymbol{\lambda},\mu)$并不是对格拉格朗日目标的二阶惩罚，而是对该目标的一个近似函数的二阶惩罚。

和$Q(\mathbf{x};\mu)$相比，$L_A(\mathbf{x};\boldsymbol{\lambda},\mu)$可极大减小$c_i(\tilde{\mathbf{x}})$的违反程度。为说明这一点，式 11.91的优化值$\tilde{\mathbf{x}}$需要满足：

$$\nabla L_A(\tilde{\mathbf{x}};\boldsymbol{\lambda},\mu) = \nabla f(\tilde{\mathbf{x}}) - \sum_{i\in\mathscr{E}}[\lambda_i - \mu c_i(\tilde{\mathbf{x}})]\nabla c_i(\tilde{\mathbf{x}}) = \mathbf{0}. \tag{11.92}$$

同时，对格拉格朗日目标$L(\mathbf{x},\boldsymbol{\lambda})$进行优化时，有：

$$\nabla L(\mathbf{x}^*,\boldsymbol{\lambda}^*) = \nabla f(\mathbf{x}^*) - \sum_{i\in\mathscr{E}} \lambda_i^* \nabla c_i(\mathbf{x}^*) = \mathbf{0}. \tag{11.93}$$

比较式 11.92和式 11.93，可知如果令λ_i和λ^*有如下关系：

$$\lambda_i - \mu c_i(\tilde{\mathbf{x}}) = \lambda_i^* \quad \forall i \in \mathscr{E} \tag{11.94}$$

则对$L_A(\mathbf{x};\boldsymbol{\lambda},\mu)$优化将得到局部最优解$\mathbf{x}^*$，此时对约束的违反程度为：

$$c_i(\tilde{\mathbf{x}}) = -\frac{1}{\mu}(\lambda_i^* - \lambda_i) \quad \forall i \in \mathscr{E}.$$

显然，只要$\boldsymbol{\lambda}$与$\boldsymbol{\lambda}^*$足够接近，$c(\hat{\mathbf{x}})$对约束的违反程度将接近于零，而不必μ的取无穷值。因此，我们希望$\boldsymbol{\lambda}$应尽可能向$\boldsymbol{\lambda}^*$靠拢。然而，计算$\boldsymbol{\lambda}^*$并不容易（否则直接优化$L(\mathbf{x},\boldsymbol{\lambda}^*)$即可得到$\mathbf{x}^*$）。式 11.94给我们提供了一种迭代方法：

$$\lambda_i^{k+1} = \lambda_i^k - \mu_k c_i(\hat{\mathbf{x}}_k) \quad \forall i \in \mathscr{E}, \tag{11.95}$$

即通过迭代使$\boldsymbol{\lambda}^{k+1}$接近$\boldsymbol{\lambda}^*$。这一迭代算法称为增广拉格朗日方法（Augmented Lagrangian Method）。算法 3给出包含等式约束的增广拉格朗日算法的过程。

```
Initialization: μ_0 > 0; λ^0;
for k:=1,2,... do
    Solve x̂_k by min_x L_A(x; λ^k, μ_k);
    if Converged(x̂_k) then
        Return x̂_k;
    end
    else
        λ^{k+1} = λ^k − μ_k c(x̂_k);
        Set μ_{k+1} ≥ μ_k;
    end
end
```

Algorithm 3: 包含等式约束的增广拉格朗日算法。

11.3.5.3 增广拉格朗日方法：不等约束

增广拉格朗日方法同样可用于不等约束，设其任务如下：

$$\min f(\mathbf{x}) \quad s.t.\ \ c_i(\mathbf{x}) \geq 0, i \in \mathscr{I}, \tag{11.96}$$

对应的拉格朗日目标函数为：

$$L(\mathbf{x},\boldsymbol{\lambda}) = f(\mathbf{x}) - \sum_{i\in\mathscr{I}} \lambda_i c_i(\mathbf{x}) \quad s.t.\ \ \boldsymbol{\lambda} \geq \mathbf{0}. \tag{11.97}$$

注意即使写成拉格朗日目标函数，其中的变量$\boldsymbol{\lambda}$也是受约束的。我们想办法将所有约束写在目标函数中。一种方法是定义如下目标函数：

$$F(\mathbf{x}) = \max_{\boldsymbol{\lambda}\geq\mathbf{0}} \left\{ f(\mathbf{x}) - \sum_{i\in\mathscr{I}} \lambda_i c_i(\mathbf{x}) \right\}.$$

容易验证，当$\mathbf{x}$处于合法域时，有$F(\mathbf{x}) = f(\mathbf{x})$，否则有$F(\mathbf{x}) = \infty$。因此，只需对$F(\mathbf{x})$进行最小化，即可得到$f(\mathbf{x})$在约束条件下的最优解。但$F(\mathbf{x})$不是连续的，因而不适合迭代求解。一种方法是对这一函数进行平滑化，引入一个对$\boldsymbol{\lambda}^*$的估计$\boldsymbol{\lambda}^k$，使得在对$\boldsymbol{\lambda}$做最大化时不能离$\boldsymbol{\lambda}^k$过远。具体形式如下：

$$\hat{F}(\mathbf{x}) = \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \left\{ f(\mathbf{x}) - \sum_{i \in \mathscr{I}} \lambda_i c_i(\mathbf{x}) - \frac{1}{2\mu} \sum_{i \in \mathscr{I}} (\lambda_i - \lambda_i^k)^2 \right\}. \tag{11.98}$$

上式中，如果$\boldsymbol{\lambda}^k = \boldsymbol{\lambda}^*$，则可直接优化得到$\mathbf{x}^*$，否则可计算$\boldsymbol{\lambda}^{k+1}$，使之更加接近于$\boldsymbol{\lambda}^*$，这正是增广拉格朗日方法的基本思路。事实上上式中优化的$\hat{\lambda}_i$是可以直接计算出来的，结果如下：

$$\hat{\lambda}_i = \begin{cases} 0 & if \quad -c_i(\mathbf{x}) + \lambda_i^k / \mu \leq 0 \\ \lambda_i^k - \mu c_i(\mathbf{x}) & otherwise \end{cases}. \tag{11.99}$$

取$\boldsymbol{\lambda}^{k+1} = \hat{\boldsymbol{\lambda}}$，即可得到对$\boldsymbol{\lambda}^*$更好的近似值。由此，我们得到增广拉格朗日方法在不等约束下的算法形式，如算法 4所示。

```
Initialization: μ_0 > 0; λ^0;
for k:=1,2,... do
    Solve x̂_k by min_x F̂(x, λ^k, μ_k);
    if Converged(x̂_k) then
        Return x̂_k;
    end
    else
        Set λ^{k+1} by Eq.11.99;
        Set μ_{k+1} ≥ μ_k;
    end
end
```

Algorithm 4: 包含不等约束的增广拉格朗日算法。

11.3.5.4 SQP

顺序二阶规划（Sequential Quadratic Programming）是非线性约束问题的另一种高效解法，其基本思路是对拉格朗日目标函数进行局部二阶近似，并对约束做局部一阶近似，由此计算局部最优值$\mathbf{x}_k$和对应的拉格朗日乘子$\boldsymbol{\lambda}_k$。

为简便起见，依然从等式约束开始讨论。设优化问题如下：

$$\min f(\mathbf{x}) \;\; s.t. \;\; \mathbf{c}(\mathbf{x}) = 0, \tag{11.100}$$

其拉格朗日目标函数为：

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{x}). \tag{11.101}$$

对上式进行优化，优化的$(\mathbf{x}, \boldsymbol{\lambda})$应满足：

$$\mathbf{F}(\mathbf{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \nabla f(\mathbf{x}) - \mathbf{A}(\mathbf{x})^T \boldsymbol{\lambda} \\ c(\mathbf{x}) \end{bmatrix} = \mathbf{0}, \tag{11.102}$$

其中$\mathbf{A}(\mathbf{x}) = [\nabla c_1(\mathbf{x}) \; ... \; \nabla c_m(\mathbf{x})]^T$为在约束上的梯度。对上式可用牛顿法求解。设当前解为$(\mathbf{x}_k, \boldsymbol{\lambda}_k)$，且记$\mathbf{A}_k = \mathbf{A}(\mathbf{x}_k), L_K = L(\mathbf{x}_k, \boldsymbol{\lambda}_k)$，则有：

$$\begin{bmatrix} \nabla^2 L_k & -\mathbf{A}_k^T \\ \mathbf{A}_k & 0 \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x} \\ \Delta \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\nabla f(\mathbf{x}_k) + \mathbf{A}_k^T \boldsymbol{\lambda}_k \\ -\mathbf{c}(\mathbf{x}_k). \end{bmatrix} \tag{11.103}$$

可以证明，如果$\mathbf{A}_k$是行满秩的，且$\nabla^2 L(\mathbf{x}, \boldsymbol{\lambda})$在限制条件的切线方向是正定的，则式 11.103有确定解。得到$\Delta \mathbf{x}$和$\Delta \boldsymbol{\lambda}$后，即可在新位置计算$\nabla^2 L_k$和$\mathbf{A}_k$。

上述推导过程用了牛顿方法，而该方法基于二阶近似，因此上面公式事实上是对$L(\mathbf{x}, \boldsymbol{\lambda})$进行二阶近似的优化过程。为更清楚看到这一点，我们将$L(\mathbf{x}, \boldsymbol{\lambda})$在$(\mathbf{x}_k, \boldsymbol{\lambda}_k)$处做二阶近似，并对限制条件$\mathbf{c}(\mathbf{x})$在$\mathbf{x}_k$ 处做一阶近似：

$$L(\mathbf{p}) \approx L_k + \mathbf{p}^T \nabla L_k + \frac{1}{2} \mathbf{p}^T \nabla^2 L_k \mathbf{p}$$

$$\mathbf{c}(\mathbf{p}) \approx \mathbf{c}_k + \mathbf{A}_k \mathbf{p}$$

原优化问题转化为局部QP问题：

$$\min_{\mathbf{p}} L(\mathbf{x}_k + \mathbf{p}, \boldsymbol{\lambda}_k) \approx L_k + \mathbf{p}^T \nabla L_k + \frac{1}{2} \mathbf{p}^T \nabla^2 L_k \mathbf{p} \quad s.t. \;\; \mathbf{A}_k \mathbf{p} + \mathbf{c}_k = 0.$$

当$\nabla^2 L_k$是正定矩阵时，上述QP问题有唯一解$(\mathbf{p}_k, \boldsymbol{\ell}_k)$，其中$\boldsymbol{\ell}_k$是拉格朗日因子。这一解对应的KKT条件为：

$$\nabla^2 L_k \mathbf{p}_k + \nabla L_k - \mathbf{A}_k^T \boldsymbol{\ell}_k = 0$$

$$\mathbf{A}_k \mathbf{p} + \mathbf{c}_k = 0$$

由于$\mathbf{A}_k \mathbf{p} + \mathbf{c}_k = \mathbf{0}$，$\nabla L_k^T \mathbf{p} = \nabla f_k^T \mathbf{p}$，因此有：

$$\nabla^2 L_k \mathbf{p}_k + \nabla f_k - \mathbf{A}_k^T \boldsymbol{\ell}_k = \mathbf{0}.$$

将上式写成如下形式：

$$\begin{bmatrix} \nabla^2 L_k & -\mathbf{A}_k^T \\ \mathbf{A}_k & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{p}_k \\ \boldsymbol{\ell}_k \end{bmatrix} = \begin{bmatrix} -\nabla f(\mathbf{x}_k) \\ -\mathbf{c}(\mathbf{x}_k)\lambda \end{bmatrix} \tag{11.104}$$

和式11.103相对比可知：

$$-\mathbf{A}_k^T \Delta\boldsymbol{\lambda} - \mathbf{A}_k^T \boldsymbol{\lambda}_k = -\mathbf{A}_k^T \boldsymbol{\ell}_k$$

因此有：

$$\boldsymbol{\ell}_k = \boldsymbol{\lambda}_k + \Delta\boldsymbol{\lambda} = \boldsymbol{\lambda}_{k+1}$$

由此可知基于牛顿法得到的更新公式 11.103事实上和对$L(\mathbf{x},\boldsymbol{\lambda})$基于当前解$(\mathbf{x}_k,\boldsymbol{\lambda}_k)$做对$\mathbf{x}$的二次近似，并对约束$c(\mathbf{x})$做一阶近似得到的结果是一致的。由此，我们可以得到包含等式约束的SQP算法，如算法 5。

```
1 Initialization: x_0; λ_0;
2 for k:=1,2,... do
3 |   Compute f_k, ∇f_k, ∇²L_k, c_k, A_k;
4 |   Compute (p_k, ℓ_k) by solving 11.104;
5 |   x_{k+1} = x_k + p_k;
6 |   λ_{k+1} = ℓ_k
7 end
```

Algorithm 5: 包含等式约束SQP算法。

上述对拉格朗日目标函数和约不做一阶或二阶近似的思路具有重要意义，因为只要实现这一近似，即可用在二阶规划一节中讨论的高效算法求解。这一思路也可以帮助我们将SQP扩展到包含非等约束的情况。设有如下非线性优化问题：

$$\min f(\mathbf{x}) \;\; s.t. \;\; c_i(\mathbf{x}) = 0, i \in \mathscr{E}; c_i(\mathbf{x}) \geq 0, i \in \mathscr{I}.$$

对上式的拉格朗日目标函数做二阶近似，对条件做一阶近似，有：

$$\begin{aligned} &\min_{\mathbf{p}} L(\mathbf{x}_k+\mathbf{p},\boldsymbol{\lambda}_k) \approx L_k + \mathbf{p}^T\nabla L_k + \tfrac{1}{2}\mathbf{p}^T\nabla^2 L_k\mathbf{p} \\ &s.t.\ \ \Delta c_i(\mathbf{x}_k)^T\mathbf{p} + c_i(\mathbf{x}_k) = 0, i\in\mathscr{E}; \\ &\Delta c_i(\mathbf{x}_k)^T\mathbf{p} + c_i(\mathbf{x}_k) \geq 0, i\in\mathscr{I}. \end{aligned} \tag{11.105}$$

上式是一个带线性不等约束的二阶优化问题，可以用任何一种二阶规划算法求解，得到优化的$\mathbf{p}_k$和$\boldsymbol{\ell}_k$，即可计算$\mathbf{x}_{k+1}$和$\boldsymbol{\lambda}_{k+1}$。因此，算法 5所示的流程可直接扩展到包含不等约束的情况，只需基于式 11.105表示的QP任务计算$(\mathbf{p}_k,\boldsymbol{\ell}_k)$。

值得说明的是，SQP只是一个简单的计算框架，在实际实现时需要考虑各种复杂问题，特别SQP基于局部一阶和二阶近似，由此得到的局部解未必在合法域中。因而可能需要采用线性搜索、信任域等方法来保证局部解的合理性，也可能需要评估这些局部解对约束条件的满足程度，以此决定是否接受该解。

11.4 本章小结

在机器学习中，绝大多数模型的目标函数非常复杂，如何对这些目标函数进行优化以完成模型训练是一个重要课题。本章介绍了机器学习中常用的几种优化方法。总体来说，这些优化方法可分为无约束优化和带约束优化两种。对于无约束优化问题，一般有两种优化策略：线性搜索和置信域优化。前者首先找到一个合理的优化方向，再确定在该方向的步长；后者首先确定一个信任域，在这一信任域内设计足够精确的近似函数，并对该近似函数进行优化。这两种策略事实上都基于同一种思路：对原函数进行一阶或二阶近似，前者基于目标函数的局部梯度信息，典型的如SGD方法，后者基于目标函数的局部曲率信息，典型的如Newton方法。二阶近似显然更加精确，但计算量更高。拟牛顿算法，如BFGS，SR1等用一阶信息的变化模拟二阶信息，通常效率更高，适合较大规模的优化任务。泰勒展开是无约束优化任务的核心。

对于有约束任务，拉格朗日乘子法是所有算法的核心。这一方法的基本思路是将带约束任务其转化成无约束任务。对于只包含等式约束的任务，拉格朗日乘子法可将有约束任务完全转化成无约束任务；对于包含不等约束的任务，用拉格朗日乘子法依然会有对变量的非负约束存在。为了有效处理这一约束，研究者提出激发集（Active Set）方法和内点法（Interior）两种方

法。激发集法构造不等约束的激发约束，沿这些约束确定的子区域进行优化；内点法解拉格朗日目标函数的KKT条件，通过迭代逐渐接近KKT的等式条件的同时，保证不等约束得到满足。激发集法和内点法对于一阶规划任务和二阶规划任务都有高效实现方法。对于一般带约束的非线性规划任务，可以在对目标函数进行优化时加入对违反约束的惩罚项，并逐渐加大惩罚力度，以满足约束的要求；为提高约束在优化过程中的限制能力，在上述算法中可以加入近似拉格朗日乘子项，即增广拉格朗日方法（Augmented Lagrangian）。对一般带约束任务进行优化的另一种思路是对目标函数和约束条件进行局部一阶或二阶近似，从而可利用一阶规划或二阶规划中的高效方法求局部近似解。基于局部近似解，可逐渐趋近局部最优解。典型的如SQP算法。

本章讨论的优化方法只是众多优化方法中极少的一部分，但是我们相信这些已经可以让我们对优化问题有个初步概念，并对不同优化算法的对比优势有了初步印象。事实上，机器学习研究者更侧重对实际任务的合理建模并选择适用于该模型的高效优化算法。至于算法本身，数值优化专家们已经做了非常深入研究，也有很多软件资源可利用，机器学习研究应该首先利用这些既有成果以加快研究步伐。

11.5 相关资源

- 本章绝大部分内容参考了Jorge Nocedal和Stephen J. Wright的著作《Numerical optimization》 [8]。
- 关于基础优化方法更详细的说明可参考 [5, 1, 3]。
- 关于凸优化问题可参考Boyd等人的著作《Convex optimization》 [4]。Boyd还提供了相应的Matlab计算包CVX [6]。
- Nesterov等人2013年的著作《Introductory lectures on convex optimization: A basic course》 [7]也是学习凸优化的不错资料。
- 关于带约束优化，可参考Bertsekas2014年的著作《Constrained optimization and Lagrange multiplier methods》 [2]。

References

[1] Bertsekas DP (1999) Nonlinear programming. Athena scientific Belmont

[2] Bertsekas DP (2014) Constrained optimization and Lagrange multiplier methods. Academic press

[3] Borwein J, Lewis AS (2010) Convex analysis and nonlinear optimization: theory and examples. Springer Science & Business Media

[4] Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge university press

[5] Gill PE, Murray W, Wright MH (1981) Practical optimization

[6] Grant M, Boyd S, Ye Y (2008) Cvx: Matlab software for disciplined convex programming

[7] Nesterov Y (2013) Introductory lectures on convex optimization: A basic course, vol 87. Springer Science & Business Media

[8] Nocedal J, Wright SJ (2006) Numerical optimization 2nd

[9] Strang G (1991) Calculus. WELLESLEY-Cambridge Press, URL https://ocw.mit.edu/ans7870/resources/Strang/Edited/Calculus/Calculus.pdf