最优化计算方法（简化版）

刘浩洋、户将、李勇锋、文再文编著

前言

最优化计算方法是运筹学、计算数学、机器学习和数据科学与大数据技术等专业的一门核心课程。最优化问题通常需要对实际需求进行定性和定量分析，建立恰当的数学模型来描述该问题，设计合适的计算方法来寻找问题的最优解，探索研究模型和算法的理论性质，考察算法的计算性能等。最优化算法广泛应用于科学与工程计算、数据科学、机器学习、人工智能、图像和信号处理、金融和经济、管理科学等众多领域。本书将介绍最优化的基本概念、典型案例、基本算法和理论，培养学生解决实际问题的能力。

本书可作为数学优化、运筹学、计算数学、机器学习、人工智能、计算机科学和数据科学等专业的本科生、研究生和相关研究人员的教材或参考书目。通过本书的学习，希望读者能掌握最优化的基本概念、最优性理论、一些典型的最优化问题（如凸优化，无约束优化，约束优化，复合优化，等等）的建模或判别、相关优化问题的基本计算方法、能学会调用基于 MATLAB 或 Python 等语言的典型优化软件程序求解一些标准的优化问题，可以灵活运用所讲授的算法和理论求解一些非标准的优化问题，并锻炼对将实际问题建立合适最优化模型、选择合适的现有软件包和算法、遇到没有现成算法自己实现简单算法等能力。

考虑到不同层次的需求，本书另有详细版（书名：《最优化：建模、算法与理论》），主要区别是详细版中包含一些复杂的概念、详细的例子和证明。在第一章简要介绍最优化基本概念之后，本书从四个方面进行讲述。

- **基础知识**：第二章介绍最优化建模和算法中经常需要使用的一些基础知识，包括范数、导数、凸集、凸函数、次梯度、共轭函数等。

- **优化建模**：第三章给出了最优化问题的一些典型分类和判别技巧，如线性规划、半定规划、最小二乘问题、复合优化、矩阵优化、随机优化等。一个实际问题根据其侧重点可以由不同的优化模型来描述，一种

优化模型也可以对应很多不同的实际应用。

- **最优性理论**：第四章介绍最优性理论，包括最优解的存在性、无约束可微问题、无约束不可微问题、带约束优化问题和凸优化问题的一阶或二阶最优性条件、对偶理论、带广义不等式约束（如半定规划问题）的对偶理论。

- **最优化算法**：第五章介绍无约束优化算法，包括线搜索方法、梯度类算法、次梯度算法、牛顿（Newton）类算法、拟牛顿类算法、信赖域算法、非线性最小二乘问题算法。第六章介绍约束优化算法，包括罚函数法、增广拉格朗日（Lagrange）函数法及其在典型凸优化问题的原始问题和对偶问题上的具体应用。第七章介绍复合优化算法，包括近似点梯度法、Nesterov 加速算法、近似点算法、分块坐标下降法、对偶算法、交替方向乘子法、随机优化算法。

本书主要概念配有详细的例子来解释，主要优化算法的介绍包含算法描述、应用举例和收敛性分析三个方面。在算法描述方面，本书侧重于算法的基本思想和直观解释；在应用举例方面，针对几乎所有算法写出了其在稀疏优化或逻辑回归等典型问题中的具体形式和求解过程，给出了最优性度量与迭代步数关系等数值结果。相关程序也可以从作者主页下载，读者可方便地比较各种算法的特点。

本书各部分内容的难易程度有些差异，理论和算法涉及的基础知识也有较大差异，比如向量导数、凸集、凸函数、线性代数等在低年级课程中大多已经覆盖，但是矩阵函数及其导数、共轭函数、次梯度等可能讲述很少。因此讲授或阅读时可以根据具体情况进行选择，不一定要按照章节的顺序进行。例如次梯度、无约束不可微问题的最优性理论和次梯度算法等涉及非光滑函数的基础部分可以考虑放在光滑函数的梯度类算法之后再讲授或阅读。

最优化理论与算法内涵十分丰富，本书涉及的各方面仍然比较初步和浅略。更全面的应用场景，更深入的理论探讨和更详细的算法设计需读者进一步查阅相关章节给出的参考文献。由于篇幅限制，有很多重要内容没有讲述，如连续优化里的共轭梯度算法、逐步二次规划、无导数优化、线性规划单纯形法和内点法、二次锥规划和半定规划的内点法、非线性规划的内点法等。本书也没有讲述带微分方程约束优化、流形约束优化、鲁棒优化、整数规划、组合优化、次模优化、动态规划等应用广泛的知识，感兴趣的读者可以阅读相关文献。

诚挚感谢袁亚湘院士多年来的精心指导和悉心关怀，对本书的规划和内容给予的宝贵意见。特别感谢张平文院士、马志明院士、徐宗本院士等专家对本书的指导和支持。非常感谢北京大学北京国际数学研究中心和数学科学学院、国家自然科学基金、北京智源人工智能研究院等对课题组的长期资助和支持。

本书写作参考了袁亚湘院士和孙文瑜教授的《最优化理论与方法》，Jorge Nocedal 教授和 Stephen Wright 教授的 *Numerical Optimization*，Stephen Boyd 教授和 Lieven Vandenberghe 教授的 *Convex Optimization* 等经典教材。Lieven Vandenberghe 教授在加州大学洛杉矶分校多门课程的讲义对本书的整理帮助很大。也特别感谢加州大学洛杉矶分校印卧涛教授慷慨分享稀疏优化、交替方向乘子法、坐标下降法等很多方面的内容。

本书内容在北京大学数学科学学院多次开设的“凸优化”和“大数据分析中的算法”课程中使用，感谢课题组同学在初稿整理方面的支持，如刘普凡在内容简介，金泽宇在数值代数基础和 Nesterov 加速算法，许东在数学分析基础，杨明瀚在无约束光滑函数优化方法，柳伊扬在无约束非光滑函数优化算法，柳昊明在近似点梯度法，刘德斌在罚函数法，赵明明在对偶函数方法，王金鑫在交替方向乘子法及其变形，陈铖和谢中林在书稿后期整理等方面的帮助。同时也感谢高等教育出版社刘荣编辑精心细致的校稿和修改。

限于作者的知识水平，书中恐有不妥之处，恳请读者不吝批评和指正。

刘浩洋、户将、李勇锋、文再文

北京，2020 年 7 月

目录

第一章 最优化简介 **1**
1.1 最优化问题概括 1
1.2 实例：稀疏优化 3
1.3 实例：深度学习 7
1.4 最优化的基本概念 11
1.5 总结 19
习题 1 20

第二章 基础知识 **21**
2.1 范数 21
2.2 导数 24
2.3 广义实值函数 29
2.4 凸集 33
2.5 凸函数 39
2.6 共轭函数 46
2.7 次梯度 49
2.8 总结 55
习题 2 56

第三章 典型优化问题 **59**
3.1 线性规划 59
3.2 最小二乘问题 61
3.3 复合优化问题 64
3.4 随机优化问题 66
3.5 半定规划 68

3.6 矩阵优化 71
3.7 优化模型语言 73
3.8 总结 74
习题 4 75

第四章 最优性理论 **79**
4.1 最优化问题解的存在性 79
4.2 无约束可微问题的最优性理论 81
4.3 无约束不可微问题的最优性理论 85
4.4 对偶理论 87
4.5 一般约束优化问题的最优性理论 97
4.6 带约束凸优化问题的最优性理论 106
4.7 约束优化最优性理论应用实例 109
4.8 总结 111
习题 5 112

第五章 无约束优化算法 **117**
5.1 线搜索方法 117
5.2 梯度类算法 126
5.3 次梯度算法 137
5.4 牛顿类算法 142
5.5 拟牛顿类算法 149
5.6 信赖域算法 158
5.7 非线性最小二乘问题算法 171
5.8 总结 182
习题 6 184

第六章 约束优化算法 **187**
6.1 罚函数法 187
6.2 增广拉格朗日函数法 200
6.3 总结 215
习题 7 215

第七章 复合优化算法 **219**
7.1 近似点梯度法 219
7.2 Nesterov 加速算法 230
7.3 近似点算法 241
7.4 分块坐标下降法 250
7.5 对偶算法 257
7.6 交替方向乘子法 267
7.7 随机优化算法 289
7.8 总结 306
习题 8 308

附录 A 符号表 **311**

参考文献 **315**

索引 **329**

第一章　最优化简介

最优化问题（也称优化问题）泛指定量决策问题，主要关心如何对有限资源进行有效分配和控制，并达到某种意义上的最优. 它通常需要对需求进行定性和定量分析，建立恰当的数学模型来描述该问题，设计合适的计算方法来寻找问题的最优解，探索研究模型和算法的理论性质，考察算法的计算性能等. 由于很多数学问题难以直接给出显式解，最优化模型就成为人们最常见的选择，计算机的高速发展也为最优化方法提供了有力辅助工具. 因此最优化方法被广泛应用于科学与工程计算、金融与经济、管理科学、工业生产、图像与信号处理、数据分析与人工智能、计算物理与化学等众多领域.

本章将介绍最优化问题的一般形式和一些重要的基本概念，并通过实际应用中的例子让读者更加直观地理解最优化问题.

1.1　最优化问题概括

1.1.1　最优化问题的一般形式

最优化问题一般可以描述为

$$\begin{aligned} \min \quad & f(x), \\ \text{s.t.} \quad & x \in \mathcal{X}, \end{aligned} \tag{1.1.1}$$

其中 $x = (x_1, x_2, \cdots, x_n)^{\mathrm{T}} \in \mathbb{R}^n$ 是**决策变量**，$f\colon \mathbb{R}^n \to \mathbb{R}$ 是**目标函数**，$\mathcal{X} \subseteq \mathbb{R}^n$ 是**约束集合**或**可行域**，可行域包含的点称为**可行解**或**可行点**. 记号 s.t. 是“subject to”的缩写，专指约束条件. 当 $\mathcal{X} = \mathbb{R}^n$ 时，问题 (1.1.1) 称为无约束优化问题. 集合 $\mathcal{X}$ 通常可以由约束函数 $c_i(x)\colon \mathbb{R}^n \to \mathbb{R}, i = 1, 2, \cdots, m + l$

表达为如下具体形式：

$$\mathcal{X} = \{x \in \mathbb{R}^n \mid c_i(x) \leqslant 0, \quad i = 1,2,\cdots,m, \\ c_i(x) = 0, \quad i = m+1, m+2 \cdots, m+l\}.$$

在所有满足约束条件的决策变量中，使目标函数取最小值的变量 x^* 称为优化问题 (1.1.1) 的最优解，即对任意 $x \in \mathcal{X}$ 都有 $f(x) \geqslant f(x^*)$. 如果我们求解在约束集合 $\mathcal{X}$ 上目标函数 $f(x)$ 的最大值，则问题 (1.1.1) 的 “min” 应相应地替换为 “max”. 注意到在集合 $\mathcal{X}$ 上，函数 f 的最小（最大）值不一定存在，但是其下（上）确界 “inf f(sup f)” 总是存在的. 因此，当目标函数的最小 (最大）值不存在时，我们便关心其下（上）确界，即将问题 (1.1.1) 中的 “min(max)” 改为 “inf(sup)”. 为了叙述简便，问题 (1.1.1) 中 x 为 $\mathbb{R}^n$ 空间中的向量. 实际上，根据具体应用和需求，x 还可以是矩阵、多维数组或张量等，本书介绍的很多理论和算法可以相应推广.

> 由于本书涉及较多公式，请读者根据上下文区分公式中的标量、向量、矩阵. 在不加说明的情况下，向量一般用小写英文字母或希腊字母表示，矩阵一般用大写英文字母或希腊字母表示. 公式中的标量可能使用多种记号，需要根据上下文确定. 读者也可参考附录 A 符号表.

1.1.2 最优化问题的类型与应用背景

最优化问题 (1.1.1) 的具体形式非常丰富，我们可以按照目标函数、约束函数以及解的性质将其分类. 按照目标函数和约束函数的形式来分：当目标函数和约束函数均为线性函数时，问题 (1.1.1) 称为线性规划；当目标函数和约束函数中至少有一个为非线性函数时，相应的问题称为非线性规划；如果目标函数是二次函数而约束函数是线性函数则称为二次规划；包含非光滑函数的问题称为非光滑优化；不能直接求导数的问题称为无导数优化；变量只能取整数的问题称为整数规划；在线性约束下极小化关于半正定矩阵的线性函数的问题称为半定规划，其广义形式为锥规划. 按照最优解的性质来分：最优解只有少量非零元素的问题称为稀疏优化；最优解是低秩矩阵的问题称为低秩矩阵优化. 此外还有几何优化、二次锥规划、张量优化、鲁棒优化、全局优化、组合优化、网络规划、随机优化、动态规划、带微分方程约束优化、微分流形约束优化、分布式优化等. 就具体应用而言，问题

(1.1.1) 可涵盖统计学习、压缩感知、最优运输、信号处理、图像处理、机器学习、强化学习、模式识别、金融工程、电力系统等领域的优化模型.

需要指出的是，数学建模很容易给出应用问题不同的模型，可以对应性质很不相同的问题，其求解难度和需要的算法也将差别很大. 在投资组合优化中，人们希望通过寻求最优的投资组合以降低风险、提高收益. 这时决策变量 x_i 表示在第 i 项资产上的投资额，向量 $x \in \mathbb{R}^n$ 表示整体的投资分配. 约束条件可能为总资金数、每项资产的最大（最小）投资额、最低收益等. 目标函数通常是某种风险度量. 如果是极小化收益的方差，则该问题是典型的二次规划；如果极小化风险价值 (value at risk) 函数，则该问题是混合整数规划；如果极小化条件风险价值（conditional value at risk) 函数，则该问题是非光滑优化，也可以进一步化成线性规划.

在本章后面的三节中，我们通过一些实际应用中的例子更直观、深入地理解最优化问题. 由于篇幅限制，我们通常只简要给出它们的一些典型形式，而且叙述并不严格，主要是提供这些应用的大致形式，详细的定义和描述请读者参考本书后面的章节或者相关参考文献. 而在第三章中，我们会将它们按优化问题分类. 本书的目标之一是使得读者通过学习本书的理论和算法，能用算法软件包来求解这些模型，并了解这些算法有哪些优缺点，更进一步地，使得读者能独立设计类似问题的算法.

1.2 实例：稀疏优化

考虑线性方程组求解问题：

$$Ax = b, \tag{1.2.1}$$

其中向量 $x \in \mathbb{R}^n$，$b \in \mathbb{R}^m$，矩阵 $A \in \mathbb{R}^{m\times n}$，且向量 b 的维数远小于向量 x 的维数，即 $m \ll n$. 在自然科学和工程中常常遇到已知向量 b 和矩阵 A，想要重构向量 x 的问题. 例如在信号传输过程中，希望通过接收到长度为 m 的数字信号精确地重构原始信号. 注意到由于 $m \ll n$，方程组 (1.2.1) 是欠定的，因此存在无穷多个解，重构出原始信号看似很难. 所幸的是，这些解当中大部分是我们不感兴趣的，真正有用的解是所谓的“稀疏解”，即原始信号中有较多的零元素. 如果加上稀疏性这一先验信息，且矩阵 A 以及原问题的解 u 满足某些条件，那么我们可以通过求解稀疏优化问题把 u 与方程组 (1.2.1) 的其他解区别开. 这类技术广泛应用于压缩感知（compressive sensing），即通过部分信息恢复全部信息的解决方案.

先来看一个具体的例子．在 MATLAB 环境里构造 A, u 和 b：

```
m = 128; n = 256;
A = randn(m, n);
u = sprandn(n, 1, 0.1);
b = A * u;
```

在这个例子中，我们构造了一个 128×256 矩阵 A，它的每个元素都服从高斯（Gauss）随机分布．精确解 u 只有 10% 的元素非零，每一个非零元素也服从高斯分布．这些特征可以在理论上保证 u 是方程组 (1.2.1) 唯一的非零元素最少的解，即 u 是如下 ℓ_0 范数[1]问题的最优解：

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \|x\|_0, \\ \text{s.t.} \quad & Ax = b. \end{aligned} \tag{1.2.2}$$

其中 $\|x\|_0$ 是指 x 中非零元素的个数．由于 $\|x\|_0$ 是不连续的函数，且取值只可能是整数，问题 (1.2.2) 实际上是 NP（non-deterministic polynomial）难的，求解起来非常困难．因此当 n 较大时通过直接求解问题 (1.2.2) 来恢复出原始信号 u 是行不通的．那有没有替代的方法呢？答案是有的．若定义 ℓ_1 范数：$\|x\|_1 = \sum_{i=1}^{n} |x_i|$，并将其替换到问题 (1.2.2) 当中，我们得到了另一个形式上非常相似的问题（又称 ℓ_1 范数优化问题，基追踪问题）：

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \|x\|_1, \\ \text{s.t.} \quad & Ax = b. \end{aligned} \tag{1.2.3}$$

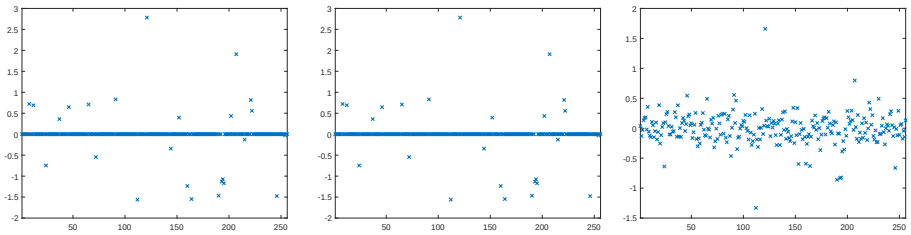
令人惊讶地是，可以从理论上证明：若 A, b 满足一定的条件（例如使用前面随机产生的 A 和 b），向量 u 也是 ℓ_1 范数优化问题 (1.2.3) 的唯一最优解．这一发现的重要之处在于，虽然问题 (1.2.3) 仍没有显式解，但与问题 (1.2.2) 相比难度已经大大降低．前面我们提到 ℓ_0 范数优化问题是 NP 难问题，但 ℓ_1 范数优化问题的解可以非常容易地通过现有优化算法得到！从这个例子不难发现，优化学科的研究能够极大程度上帮助我们攻克现有的困难问题．既然有如上令人兴奋的结果，我们是否能使用其他更容易求解的范数替代 ℓ_0 范数呢？事实并非如此．如果简单地把 ℓ_1 范数修改为 ℓ_2 范数：

[1]实际上，ℓ_0 范数不是一个范数，这里为了叙述统一而采用了这个术语，读者应当注意这个区别.

$\|x\|_2 = \left(\sum_{i=1}^{n} x_i^2\right)^{1/2}$，即求解如下优化问题：

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \|x\|_2, \\ \text{s.t.} \quad & Ax = b. \end{aligned} \tag{1.2.4}$$

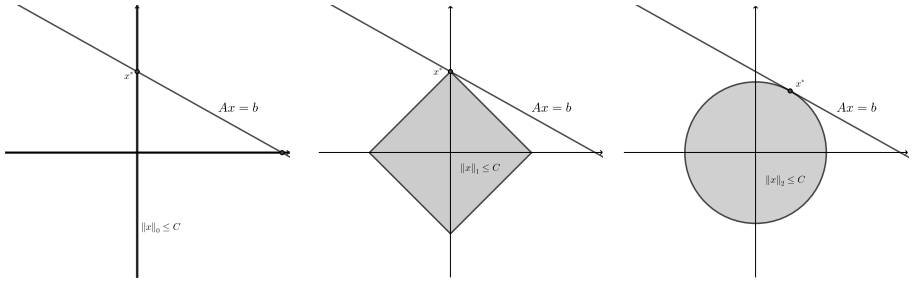
几何学的知识表明，问题 (1.2.4) 实际上就是原点到仿射集 $Ax = b$ 的投影，我们可以直接写出它的显式表达式. 但遗憾的是，u 并不是问题 (1.2.4) 的解. 事实上，图 1.1(a)-(c) 分别给出了一组随机数据下的 u，以及问题 (1.2.3) 和问题 (1.2.4) 的数值解. 可以看出图 1.1(a) 和 (b) 是完全一样的，而 (c) 则与 u 相去甚远，虽然隐约能看出数据点的大致趋势，但已经不可分辨非零元素的具体位置.



(a) 精确解 u　(b) 问题 (1.2.3) 的解　(c) 问题 (1.2.4) 的解

图 1.1　稀疏优化的例子

为什么会出现这种情况呢？这要追溯到 ℓ_0, ℓ_1, ℓ_2 范数的性质. 下面用图示的方式来直观说明为什么 ℓ_1 范数优化问题的解具有稀疏性而 ℓ_2 范数优化问题的解不具有该性质. 为了方便起见，我们在二维空间上讨论求解欠定方程组 $Ax = b$，此时 $Ax = b$ 是一条直线. 在几何上，三种优化问题实际上要找到最小的 C，使得“范数球” $\{x|\|x\| \leqslant C\}$（$\|\cdot\|$ 表示任何一种范数）恰好与 $Ax = b$ 相交. 而图 1.2 里分别展示了三种范数球的几何直观：对 ℓ_0 范数，当 $C = 2$ 时 $\{x|\|x\|_0 \leqslant C\}$ 是全平面，它自然与 $Ax = b$ 相交，而当 $C = 1$ 时退化成两条直线（坐标轴），此时问题的解是 $Ax = b$ 和这两条直线的交点；对 ℓ_1 范数，根据 C 不同 $\{x|\|x\|_1 \leqslant C\}$ 为一系列正方形，这些正方形的顶点恰好都在坐标轴上，而最小的 C 对应的正方形和直线 $Ax = b$ 的交点一般都是顶点，因此 ℓ_1 范数的解有稀疏性；对 ℓ_2 范数，当 C 取值不同时 $\{x|\|x\|_2 \leqslant C\}$ 为一系列圆，而圆有光滑的边界，它和直线 $Ax = b$ 的切点可以是圆周上的任何一点，所以 ℓ_2 范数优化问题一般不能保证解的稀疏性.



(a) ℓ_0 范数 (b) ℓ_1 范数 (c) ℓ_2 范数

图 1.2 三种范数优化问题求解示意图

问题 (1.2.3) 的理论和算法研究在 2006 年左右带来了革命性的影响. 理论上研究的课题包括什么条件下问题 (1.2.3) 的解具有稀疏性，如何改进这些条件，如何推广这些条件到其他应用. 常见的数据矩阵 A 一般由离散余弦变换、小波变换、傅里叶（Fourier) 变换等生成. 虽然这些矩阵本身并没有稀疏性，但通常具有很好的分析性质，保证稀疏解的存在性. 注意到绝对值函数在零点处不可微，问题 (1.2.3) 是非光滑优化问题. 虽然它可以等价于线性规划问题，但是数据矩阵 A 通常是稠密矩阵，甚至 A 的元素未知或者不能直接存储，只能提供 Ax 或 $A^{\mathrm{T}}y$ 等运算结果. 在这些特殊情况下，线性规划经典的单纯形法和内点法通常不太适用于求解大规模的问题 (1.2.3). 本书的一个主要目的就是根据这些问题的特点设计合适的算法进行求解. 需要强调的是，问题 (1.2.3) 主要特点是其最优解是稀疏向量，它是稀疏优化的一种典型形式.

本书还将考虑带 ℓ_1 范数正则项的优化问题

$$\min_{x\in\mathbb{R}^n} \quad \mu\|x\|_1 + \frac{1}{2}\|Ax-b\|_2^2, \tag{1.2.5}$$

其中 $\mu > 0$ 是给定的正则化参数. 问题 (1.2.5) 又称为 LASSO（least absolute shrinkage and selection operator），该问题可以看成是问题 (1.2.3) 的二次罚函数形式. 由于它是无约束优化问题，形式上看起来比问题 (1.2.3) 简单. 本书大部分数值算法都将针对问题 (1.2.3) 或问题 (1.2.5) 给出具体形式. 因此全面掌握它们的求解方法是掌握基本最优化算法的一个标志.

1.3 实例：深度学习

深度学习（deep learning）的起源可以追溯至 20 世纪 40 年代，其雏形出现在控制论中. 近十年来深度学习又重新走入了人们的视野，深度学习问题和算法的研究也经历了一次新的浪潮. 虽然卷积网络的设计受到了生物学和神经科学的启发，但深度学习目前的发展早已超越了机器学习模型中的神经科学观点. 它用相对简单的函数来表达复杂的表示，从低层特征概括到更加抽象的高层特征，让计算机从经验中挖掘隐含的信息和价值. 本节我们将通过介绍多层感知机和卷积神经网络来了解优化模型在深度学习中的应用.

1.3.1 多层感知机

多层感知机（multi-layer perceptron, MLP）也叫作深度前馈网络（deep feedforward network）或前馈神经网络（feedforward neural network），它通过已有的信息或者知识来对未知事物进行预测. 在神经网络中，已知的信息通常用数据集来表示. 数据集一般分为训练集和测试集：训练集是用来训练神经网络，从而使得神经网络能够掌握训练集上的信息；测试集是用来测试训练完的神经网络的预测准确性. 一个常见的任务是分类问题. 假设我们有一个猫和狗的图片集，将其划分成训练集和测试集（保证集合中猫和狗图片要有一定的比例）. 神经网络是想逼近一个从图片到 $\{0,1\}$ 的函数，这里 0 表示猫，1 表示狗. 因为神经网络本身的结构和大量的训练集信息，训练得到的函数与真实结果具有非常高的吻合性.

具体地，给定训练集 $D=\{\{a_1,b_1\},\{a_2,b_2\},\cdots,\{a_m,b_m\}\}$，假设数据 $a_i\in\mathbb{R}^p, b_i\in\mathbb{R}^q$. 为了方便处理模型里的偏差项，还假设 a_i 的第一个元素等于 1，即 $a_{i1}=1$. 图 1.3 给出了一种由 p 个输入单元和 q 个输出单元构成的 $(L+2)$ 层感知机，其含有一个输入层，一个输出层，和 L 个隐藏层. 该感知机的第 l 个隐藏层共有 $m^{(l)}$ 个神经元，为了方便我们用 $l=0$ 表示输入层，$l=L+1$ 表示输出层，并定义 $m^{(0)}=p$ 和 $m^{(L+1)}=q$. 设 $y^{(l)}\in\mathbb{R}^{m^{(l)}}$ 为第 l 层的所有神经元，同样地，为了能够处理每一个隐藏层的信号偏差，除输出层外，我们令 $y^{(l)}$ 的第一个元素等于 1，即 $y_1^{(l)}=1, 0\leqslant l\leqslant L$，而其余的元素则是通过上一层的神经元的值进行加权求和得到. 令参数 $x=(x^{(1)},x^{(2)},\cdots,x^{(L+1)})$ 表示网络中所有层之间的权重，其中 $x_{i,k}^{(l)}$ 是第 $(l-1)$ 隐藏层的第 k 个单元连接到第 l 隐藏层的第 i 个单元对应的权重，则在第 l 隐藏层中，第 i 个单

元（$i>1$，当 $l=L+1$ 时可取为 $i\geqslant 1$）计算输出信息 $y_i^{(l)}$ 为

$$y_i^{(l)}=t(z_i^{(l)}),\quad z_i^{(l)}=\sum_{k=1}^{m^{(l-1)}}x_{i,k}^{(l)}y_k^{(l-1)}. \tag{1.3.1}$$

这里函数 $t(\cdot)$ 称为激活函数，常见的类型有 Sigmoid 函数

$$t(z)=\frac{1}{1+\exp(-z)},$$

Heaviside 函数

$$t(z)=\begin{cases}1, & z\geqslant 0,\\ 0, & z<0,\end{cases}$$

以及 ReLU 函数

$$t(z)=\max\{0,z\}. \tag{1.3.2}$$

整个过程可以描述为

$$y^{(0)}\xrightarrow{x^{(1)}}z^{(1)}\xrightarrow{t}y^{(1)}\xrightarrow{x^{(2)}}\cdots\xrightarrow{t}y^{(L+1)}.$$

容易看出，多层感知机的每一层输出实际就是由其上一层的数值作线性组合再逐分量作非线性变换得到的．若将 $y^{(0)}$ 视为自变量，$y^{(L+1)}$ 视为因变量，则多层感知机实际上定义了一个以 x 为参数的函数 $h(a;x):\mathbb{R}^p\to\mathbb{R}^q$，这里 a 为输入层 $y^{(0)}$ 的取值．当输入数据为 a_i 时，其输出 $h(a_i;x)$ 将作为真实标签 b_i 的估计．若选择平方误差为损失函数，则我们得到多层感知机的优化模型：

$$\min_x\quad\sum_{i=1}^{m}\|h(a_i;x)-b_i\|_2^2+\lambda r(x), \tag{1.3.3}$$

其中 $r(x)$ 是正则项，用来刻画解的某些性质，如光滑性或稀疏性等；λ 称为正则化参数，用来平衡模型的拟合程度和解的性质．如果 λ 太小，那么对解的性质没有起到改善作用；如果 λ 太大，则模型与原问题相差很大，可能是一个糟糕的逼近．

1.3.2 卷积神经网络

卷积神经网络（convolutional neural network，CNN）是一种深度前馈人工神经网络，专门用来处理如时间序列数据或是图像等网格数据．CNN 在计算机视觉、视频分析、自然语言处理等诸多领域有大量成功的应用．与

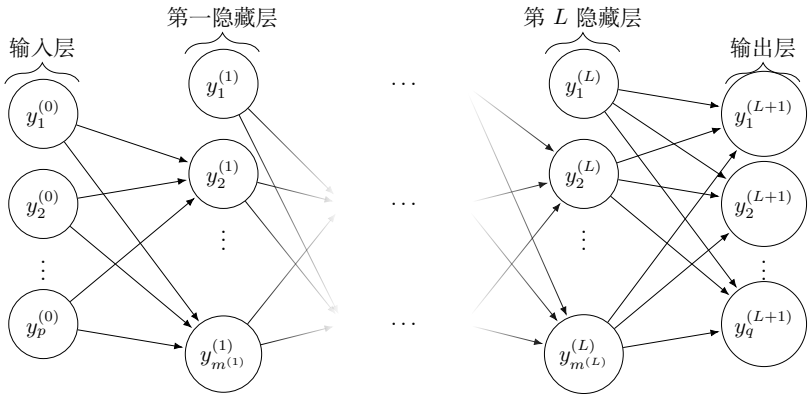


图 1.3 带 p 个输入单元和 q 个输出单位的 $(L+2)$ 层感知机的网络图，第 l 个隐藏层包含 $m^{(l)}$ 个神经元.

图 1.3 对应的全连接网络（相邻两层之间的节点都是相连或相关的）不同，卷积神经网络的思想是通过局部连接以及共享参数的方式来大大减少参数量，从而减少对数据量的依赖以及提高训练的速度. 典型的 CNN 网络结构通常由一个或多个卷积层、下采样层（subsampling）[2]和顶层的全连接层组成. 全连接层的结构与多层感知机的结构相同. 卷积层是一种特殊的网络层，它首先对输入数据进行卷积操作产生多个特征映射，之后使用非线性激活函数（比如 ReLU）对每个特征进行变换. 下采样层一般位于卷积层之后，它的作用是减小数据维数并提取数据的多尺度信息，其结果最终会输出到下一组变换.

给定一个二维图像 $\boldsymbol{I} \in \mathbb{R}^{n\times n}$ 和卷积核 $\boldsymbol{K} \in \mathbb{R}^{k\times k}$，我们定义一种简单的卷积操作 $\boldsymbol{S} = \boldsymbol{I} * \boldsymbol{K}$，它的元素是

$$\boldsymbol{S}_{i,j} = \langle \boldsymbol{I}(i:i+k-1, j:j+k-1), \boldsymbol{K} \rangle, \tag{1.3.4}$$

其中两个矩阵 X, Y 的内积是它们相应元素乘积之和，即 $\langle X, Y \rangle = \sum_{i,j} X_{ij} Y_{ij}$，$\boldsymbol{I}(i:i+k-1, j:j+k-1)$ 是矩阵 $\boldsymbol{I}$ 从位置 (i,j) 开始的一个 $k \times k$ 子矩阵. 图 1.4 给出了一个例子. 生成的结果 $\boldsymbol{S}$ 可以根据卷积核的维数、$\boldsymbol{I}$ 的边界是否填充、卷积操作时滑动的大小等相应变化.

图 1.5 给出了下采样层的一个示例. 第 l 特征层是第 $(l-1)$ 特征层的下采样层. 具体地，我们先将第 $(l-1)$ 层的每个矩阵划分成若干子矩阵，之后

[2]有时也称为“池化”（pooling）

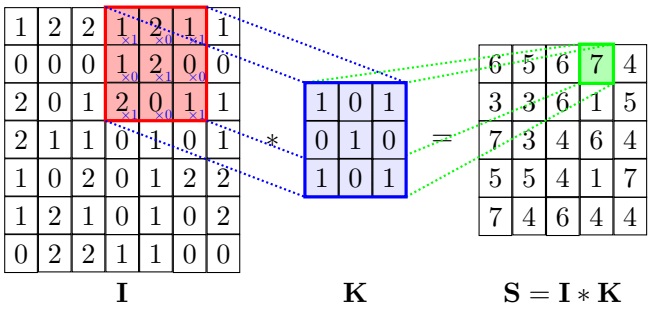


图 1.4 卷积操作

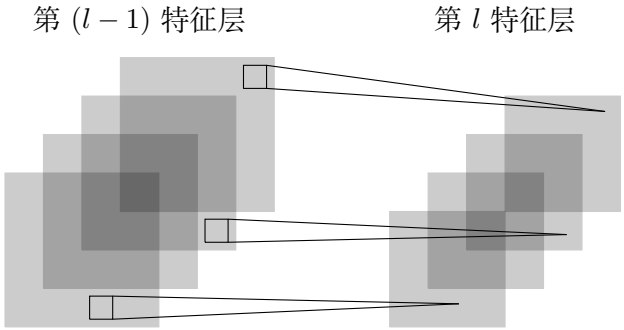


图 1.5 下采样层示例

将每个子矩阵里所有元素按照某种规则（例如取平均值或最大值）变换成一个元素. 因此，第 $(l-1)$ 特征层每个小框里所有元素的平均值或最大值就对应于第 l 特征层的一个元素. 容易看出，下采样层实际上是用一个数代表一个子矩阵，经过下采样层变换后，前一特征层矩阵的维数会进一步降低. 图 1.6 给出了一个简单的卷积神经网络示意图. 输入图片通过不同的卷积核生成的不同矩阵，再经过非线性激活函数作用后生成第 1 层的特征；第 2 层是第 1 层的下采样层；第 3 层和第 4 层又是卷积层和下采样层；第 5 层是全连接层；第 6 层为输出层. 实际的卷积神经网络可达几十层甚至更多，卷积核的大小，网络节点之间的连接方式也可以有很多变化，从而生成不一样的模型.

给定一个训练集 $D = \{\{a_1, b_1\}, \{a_2, b_2\}, \cdots, \{a_m, b_m\}\}$，其中 a_i 是训练图片，b_i 是其对应的标签. 卷积神经网络对应的优化问题的形式仍可套用 (1.3.3)，但函数 $h(a_i; x)$ 由卷积神经网络构成，而 x 是卷积神经网络的参数.

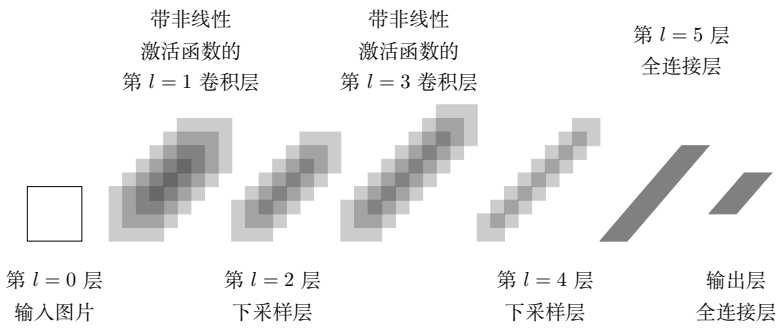


图 1.6 卷积神经网络一种示意图

1.4 最优化的基本概念

一般来说，最优化算法研究可以分为：构造最优化模型、确定最优化问题的类型和设计算法、实现算法或调用优化算法软件包进行求解. 最优化模型的构造和实际问题紧密相关，比如说，给定二维欧几里得（Euclid）空间的若干个离散点，假定它们可以通过一条直线分成两部分，也可以通过一条曲线分成两部分. 那么分别使用直线与曲线所得到的最优化模型是不同的. 在问题 (1.1.1) 中，目标函数 f 和约束函数 c_i 都是由模型来确定的. 在确定模型之后，我们需要对模型对应的优化问题进行分类. 这里，分类的必要性是因为不存在对于所有优化问题的一个统一的算法. 因此我们需要针对具体优化问题所属的类别，来设计或者调用相应的算法求解器. 最后就是模型的求解过程. 同一类优化问题往往存在着不同的求解算法. 对于具体的优化问题，我们需要充分利用问题的结构，并根据问题的需求（求解精度和速度等）来设计相应的算法. 另外，根据算法得到的结果，我们可以来判别模型构造是否合理或者进一步地改进模型. 如果构造的模型比较复杂，那么算法求解起来相对困难（时间慢或者精度差）. 此时算法分析可以帮助我们设计替代模型，以确保快速且比较精确地求出问题的解.

这三个部分的研究对于形成完备的最优化体系是必要的. 实际应用导出的各种各样的最优化模型给最优化学科不断注入新鲜的血液，对现有的优化算法进行挑战并推动其向前发展. 最优化算法的设计以及理论分析帮助实际问题建立更鲁棒稳定的模型. 模型与算法相辅相成，使得最优化学科不断发展.

1.4.1 连续和离散优化问题

最优化问题可以分为连续和离散优化问题两大类. 连续优化问题是指决策变量所在的可行集合是连续的，比如平面、区间等. 如稀疏优化问题 (1.2.2) — (1.2.5) 的约束集合就是连续的. 离散优化问题是指决策变量能在离散集合上取值，比如离散点集、整数集等. 常见的离散优化问题有整数规划，其对应的决策变量的取值范围是整数集合.

在连续优化问题中，基于决策变量取值空间以及约束和目标函数的连续性，我们可以从一个点处目标和约束函数的取值来估计该点可行邻域内的取值情况. 进一步地，可以根据邻域内的取值信息来判断该点是否最优. 离散优化问题则不具备这个性质，因为决策变量是在离散集合上取值. 因此在实际中往往比连续优化问题更难求解. 实际中的离散优化问题往往可以转化为一系列连续优化问题来进行求解. 比如线性整数规划问题中著名的分支定界方法，就是松弛成一系列线性规划问题来进行求解. 因此连续优化问题的求解在最优化理论与算法中扮演着重要的角色. 本书后续的内容也将围绕连续优化问题展开介绍.

1.4.2 无约束和约束优化问题

最优化问题的另外一个重要的分类标准是约束是否存在. 无约束优化问题的决策变量没有约束条件限制，即可行集合 $\mathcal{X} = \mathbb{R}^n$. 相对地，约束优化问题是指带有约束条件的问题. 在实际应用中，这两类优化问题广泛存在. 无约束优化问题对应于在欧几里得空间中求解一个函数的最小值点. 比如在 ℓ_1 正则化问题 (1.2.5) 中，决策变量的可行域是 $\mathbb{R}^n$，其为一个无约束优化问题. 在问题 (1.2.2) — (1.2.4) 中，可行集为 $\{x \mid Ax = b\}$，其为约束优化问题.

因为问题 (1.1.1) 可以通过将约束（$\mathcal{X} \neq \mathbb{R}^n$）罚到目标函数上转化为无约束问题，所以在某种程度上，约束优化问题就是无约束优化问题. 很多约束优化问题的求解也是转化为一系列的无约束优化问题来做，常见方式有增广拉格朗日函数法、罚函数法等. 尽管如此，约束优化问题的理论以及算法研究仍然是非常重要的. 主要原因是，借助于约束函数，我们能够更好地描述可行域的几何性质，进而更有效地找到最优解. 对于典型的约束和无约束优化模型，我们将会在本书的第三章中介绍，相应的理论以及算法会在第四—七章中给出.

1.4.3 随机和确定性优化问题

伴随着近年来人工智能的发展，随机优化问题的研究得到了长足的发展. 随机优化问题是指目标或者约束函数中涉及随机变量而带有不确定性的问题. 不像确定性优化问题中目标和约束函数都是确定的，随机优化问题中总是包含一些未知的参数. 在实际问题中，我们往往只能知道这些参数的某些估计. 随机优化问题在机器学习、深度学习以及强化学习中有着重要应用，其优化问题的目标函数是关于一个未知参数的期望的形式. 因为参数的未知性，实际中常用的方法是通过足够多的样本来逼近目标函数，得到一个新的有限和形式的目标函数. 由于样本数量往往非常大，我们还是将这个问题看作相对于指标随机变量的期望形式，然后通过随机优化方法来进行求解.

相比于确定性优化问题，随机优化问题的求解往往涉及更多的随机性. 很多确定性优化算法都有相应的随机版本. 随机性使得这些算法在特定问题上具有更低的计算复杂度或者更好的收敛性质. 以目标函数为多项求和的优化问题为例，如果使用确定性优化算法，每一次计算目标函数的梯度都会引入昂贵的复杂度. 但是对于随机优化问题，我们每次可能只计算和式中的一项或者几项，这大大减少了计算时间. 同时我们还能保证算法求解的足够精确. 具体的模型介绍会在第三章中给出. 确定性优化算法会在第五—七章中给出，随机优化算法会在第七章中介绍.

1.4.4 线性和非线性规划问题

线性规划是指问题 (1.1.1) 中目标函数和约束函数都是线性的. 当目标函数和约束函数至少有一个是非线性的，那么对应的优化问题的称为非线性规划问题. 线性规划问题在约束优化问题中具有较为简单的形式. 类似于连续函数可以用分片线性函数来逼近一样，线性规划问题的理论分析与数值求解可以为非线性规划问题提供很好的借鉴和基础.

线性规划问题的研究很早便得到了人们的关注.在 1946—1947 年，George Bernard Dantzig 提出了线性规划的一般形式并提出了至今仍非常流行的单纯形方法. 虽然单纯形方法在实际问题中经常表现出快速收敛，但是其复杂度并不是多项式的. 1979 年，Leonid Khachiyan 证明了线性规划问题多项式时间算法的存在性. 1984 年，Narendra Karmarkar 提出了多项式时间的内点法. 后来，内点法也被推广到求解一般的非线性规划问题. 目前，求解

线性规划问题最流行的两类方法依然是单纯形法和内点法.

1.4.5 凸和非凸优化问题

凸优化问题是指最小化问题 (1.1.1) 中的目标函数和可行域分别是凸函数和凸集. 如果其中有一个或者两者都不是凸的, 那么相应的最小化问题是非凸优化问题. 因为凸优化问题的任何局部最优解都是全局最优解, 其相应的算法设计以及理论分析相对非凸优化问题简单很多.

注 1.1 若问题 (1.1.1) 中的 min 改为 max, 且目标函数和可行域分别为凹函数和凸集, 我们也称这样的问题为凸优化问题. 这是因为对凹函数求极大等价于对其相反数 (凸函数) 求极小.

在实际问题的建模中, 我们经常更倾向于得到一个凸优化模型. 另外, 判断一个问题是否是凸问题也很重要. 比如, 给定一个非凸优化问题, 一种方法是将其转化为一系列凸优化子问题来求解. 此时需要清楚原非凸问题中的哪个或哪些函数导致了非凸性, 之后考虑的是如何用凸优化模型来逼近原问题. 在压缩感知问题中, ℓ_0 范数是非凸的, 原问题对应的解的性质难以直接分析, 相应的全局收敛的算法也不容易构造. 利用 ℓ_0 范数和 ℓ_1 范数在某种意义上的等价性, 我们将原非凸问题转化为凸优化问题. 在一定的假设下, 我们通过求解 ℓ_1 范数对应的凸优化问题得到了原非凸优化问题的全局最优解.

1.4.6 全局和局部最优解

在求解最优化问题之前, 先介绍最小化问题 (1.1.1) 的最优解的定义.

定义 1.1 (最优解) 对于可行点 $\bar{x}$ (即 $\bar{x} \in \mathcal{X}$), 定义如下概念:

(1) 如果 $f(\bar{x}) \leqslant f(x), \forall x \in \mathcal{X}$, 那么称 $\bar{x}$ 为问题 (1.1.1) 的**全局极小解 (点)**, 有时也称为 (全局) 最优解或最小值点;

(2) 如果存在 $\bar{x}$ 的一个 ε 邻域 $N_\varepsilon(\bar{x})$ 使得 $f(\bar{x}) \leqslant f(x), \forall x \in N_\varepsilon(\bar{x}) \cap \mathcal{X}$, 那么称 $\bar{x}$ 为问题 (1.1.1) 的**局部极小解 (点)**, 有时也称为局部最优解;

(3) 进一步地, 如果有 $f(\bar{x}) < f(x), \forall x \in N_\varepsilon(\bar{x}) \cap \mathcal{X}$, $x \neq \bar{x}$ 成立, 则称 $\bar{x}$ 为问题 (1.1.1) 的**严格局部极小解 (点)**.

如果一个点是局部极小解，但不是严格局部极小解，我们称之为**非严格局部极小解**. 在图 1.7 中，我们以一个简单的函数为例，指出了其全局与局部极小解.

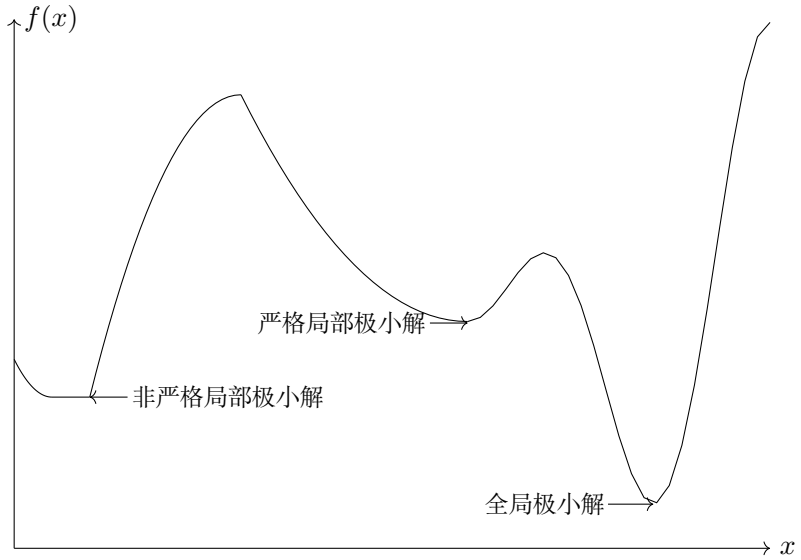


图 1.7 函数的全局极小、严格局部极小和非严格局部极小解

在问题 (1.1.1) 的求解中，我们想要得到的是其全局最优解，但是由于实际问题的复杂性，往往只能够得到其局部最优解. 在第三章中，我们将会针对具体的优化问题来分析其全局与局部最优解.

1.4.7 优化算法

在给定优化问题之后，我们要考虑如何求解. 根据优化问题的不同形式，其求解的困难程度可能会有很大差别. 对于一个优化问题，如果我们能用代数表达式给出其最优解，那么这个解称为显式解，对应的问题往往比较简单. 例如二次函数在有界区间上的极小化问题，我们可以通过比较其在对称轴上和区间两个端点处的值得到最优解，这个解可以显式地写出. 但实际问题往往是没有办法显式求解的，因此常采用迭代算法.

迭代算法的基本思想是：从一个初始点 x^0 出发，按照某种给定的规则进行迭代，得到一个序列 $\{x^k\}$. 如果迭代在有限步内终止，那么希望最后一个点就是优化问题的解. 如果迭代点列是无穷集合，那么希望该序列的极限

点（或者聚点）则为优化问题的解. 为了使算法能在有限步内终止，我们一般会通过一些收敛准则来保证迭代停在问题的一定精度逼近解上. 对于无约束优化问题，常用的收敛准则有

$$\frac{f(x^k)-f^*}{\max\{|f^*|,1\}} \leqslant \varepsilon_1, \quad \|\nabla f(x^k)\| \leqslant \varepsilon_2, \tag{1.4.1}$$

其中 $\varepsilon_1, \varepsilon_2$ 为给定的很小的正数，$\|\cdot\|$ 表示某种范数（这里可以简单理解为 ℓ_2 范数：$\|x\|_2 = \left(\sum_{i=1}^{n} x_i^2\right)^{1/2}$，第二章将会给出范数的一般定义），$f^*$ 为函数 f 的最小值（假设已知或者以某种方式估计得到）以及 $\nabla f(x^k)$ 表示函数 f 在点 x 处的梯度（光滑函数在局部最优点处梯度为零向量，第四章中会给出更多介绍）. 对于约束优化问题，还需要考虑约束违反度. 具体地，要求最后得到的点满足

$$\begin{aligned} c_i(x^k) &\leqslant \varepsilon_3,\ i = 1,2,\cdots,m, \\ |c_i(x^k)| &\leqslant \varepsilon_4,\ i = m+1, m+2, \cdots, m+l, \end{aligned}$$

其中 $\varepsilon_3, \varepsilon_4$ 为很小的正数，用来刻画 x^k 的可行性. 除了约束违反度之外，我们也要考虑 x^k 与最优解之间的距离，如 (1.4.1) 式中给出的函数值与最优值的相对误差. 由于一般情况下事先并不知道最优解，在最优解唯一的情形下一般使用某种基准算法来得到 x^* 的一个估计，之后计算其与 x^k 的距离以评价算法的性能. 因为约束的存在，我们不能简单地用目标函数的梯度来判断最优性，实际中采用的判别准则是点的最优性条件的违反度（关于约束优化的最优性条件，会在第四章中给出）.

对于一个具体的算法，根据其设计的出发点，我们不一定能得到一个高精度的逼近解. 此时，为了避免无用的计算开销，我们还需要一些停机准则来及时停止算法的进行. 常用的停机准则有

$$\frac{\|x^{k+1}-x^k\|}{\max\{\|x^k\|,1\}} \leqslant \varepsilon_5, \quad \frac{|f(x^{k+1})-f(x^k)|}{\max\{|f(x^k)|,1\}} \leqslant \varepsilon_6,$$

这里的各个 ε 一般互不相等. 上面的准则分别表示相邻迭代点和其对应目标函数值的相对误差很小. 在算法设计中，这两个条件往往只能反映迭代点列接近收敛，但不能代表收敛到优化问题的最优解.

在算法设计中，一个重要的标准是算法产生的点列是否收敛到优化问题的解. 对于问题 (1.1.1)，其可能有很多局部极小解和全局极小解，但所有全局极小解对应的目标函数值，即优化问题的最小值 f^* 是一样的. 考虑无

约束的情形，对于一个算法，给定初始点 x^0，记其迭代产生的点列为 $\{x^k\}$. 如果 $\{x^k\}$ 在某种范数 $\|\cdot\|$ 的意义下满足

$$\lim_{k\to\infty}\|x^k - x^*\| = 0,$$

且收敛的点 x^* 为一个局部（全局）极小解，那么我们称该点列收敛到局部（全局）极小解，相应的算法称为是**依点列收敛到局部（全局）极小解**的.

在算法的收敛分析中，初始迭代点 x^0 的选取也尤为重要. 比如一般的牛顿法，只有在初始点足够接近局部（全局）最优解时，才能收敛. 但是这样的初始点的选取往往比较困难，此时我们更想要的是一个从任何初始点出发都能收敛的算法. 因此优化算法的研究包括如何设计全局化策略，将已有的可能发散的优化算法修改得到一个新的全局收敛到局部（全局）最优解的算法. 比如通过采用合适的全局化策略，我们可以修正一般的牛顿法使得修改后的算法是全局收敛到局部（全局）最优解的.

进一步地，如果从任意初始点 x^0 出发，算法都是依点列收敛到局部（全局）极小解的，我们称该算法是**全局依点列收敛到局部（全局）极小解**的. 相应地，如果记对应的函数值序列为 $\{f(x^k)\}$，我们还可以定义算法的**（全局）依函数值收敛到局部（全局）极小值**的概念. 对于凸优化问题，因为其任何局部最优解都为全局最优解，算法的收敛性都是相对于其全局极小而言的. 除了点列和函数值的收敛外，实际中常用的还有每个迭代点的最优性条件（如无约束优化问题中的梯度范数，约束优化问题中的最优性条件违反度等等）的收敛.

对于带约束的情形，给定初始点 x^0，算法产生的点列 $\{x^k\}$ 不一定是可行的（即 $x^k \in \mathcal{X}$ 未必对任意 k 成立）. 考虑到约束违反的情形，我们需要保证 $\{x^k\}$ 在收敛到 x^* 的时候，其违反度是可接受的. 除此要求之外，算法的收敛性的定义和无约束情形相同.

在设计优化算法时，我们有一些基本的准则或技巧. 对于复杂的优化问题，基本的想法是将其转化为一系列简单的优化问题（其最优解容易计算或者有显式表达式）来逐步求解. 常用的技巧有：

(1) **泰勒（Taylor）展开.** 对于一个非线性的目标或者约束函数，我们通过其泰勒展开用简单的线性函数或者二次函数来逼近，从而得到一个简化的问题. 因为该简化问题只在小邻域内逼近原始问题，所以我们需要根据迭代点的更新来重新构造相应的简

化问题.

(2) **对偶.** 每个优化问题都有对应的对偶问题. 特别是凸的情形，当原始问题比较难解的时候，其对偶问题可能很容易求解. 通过求解对偶问题或者同时求解原始问题和对偶问题，我们可以简化原始问题的求解，从而设计更有效的算法.

(3) **拆分.** 对于一个复杂的优化问题，我们可以将变量进行拆分，比如 $\min_x h(x)+r(x)$，可以拆分成

$$\min_{x,y} \quad h(x)+r(y), \quad \text{s.t.} \quad x=y.$$

通过引入更多的变量，我们可以得到每个变量的简单问题（较易求解或者解有显式表达式），从而通过交替求解等方式来得到原问题的解.

(4) **块坐标下降.** 对于一个 n 维空间（n 很大）的优化问题，我们可以通过逐步求解分量的方式将其转化为多个低维空间中的优化问题. 比如，对于 $n=100$，我们可以先固定第 2—100 个分量，来求解 x_1；接着固定下标为 1,3—100 的分量来求解 x_2；依次类推.

关于这些技巧的具体应用，读者可以进一步阅读本书中的算法部分.

对于同一个优化问题，其求解算法可以有很多. 在设计和比较不同的算法时，另一个重要的指标是算法的渐进收敛速度. 我们以点列的 **Q-收敛速度**（Q 的含义为 “quotient”）为例（函数值的 Q-收敛速度可以类似地定义）. 设 $\{x^k\}$ 为算法产生的迭代点列且收敛于 x^*，若对充分大的 k 有

$$\frac{\|x^{k+1}-x^*\|}{\|x^k-x^*\|} \leqslant a, \quad a\in(0,1),$$

则称算法（点列）是 **Q-线性收敛**的；若满足

$$\lim_{k\to\infty} \frac{\|x^{k+1}-x^*\|}{\|x^k-x^*\|} = 0,$$

称算法（点列）是 **Q-超线性收敛**的；若满足

$$\lim_{k\to\infty} \frac{\|x^{k+1}-x^*\|}{\|x^k-x^*\|} = 1,$$

称算法（点列）是 **Q-次线性收敛**的．若对充分大的 k 满足

$$\frac{\|x^{k+1}-x^*\|}{\|x^k-x^*\|^2}\leqslant a,\quad a>0,$$

则称算法（点列）是 **Q-二次收敛**的．类似地，也可定义更一般的 Q-r 次收敛（$r>1$）．

除 Q-收敛速度外，另一常用概念是 **R-收敛速度**（R 的含义为 “root”）．以点列为例，设 $\{x^k\}$ 为算法产生的迭代点且收敛于 x^*，若存在 Q-线性收敛于 0 的非负序列 t_k 并且

$$\|x^k-x^*\|\leqslant t_k$$

对任意的 k 成立，则称算法（点列）是 **R-线性收敛**的．类似地，可定义 **R-超线性收敛**和 **R-二次收敛**等收敛速度．从 R-收敛速度的定义可以看出序列 $\{\|x^k-x^*\|\}$ 被另一趋于 0 的序列 $\{t_k\}$ 控制．当知道 t_k 的形式时，我们也称算法（点列）的收敛速度为 $\mathcal{O}(t_k)$．

与收敛速度密切相关的概念是优化算法的**复杂度** $N(\varepsilon)$，即计算出给定精度 ε 的解所需的迭代次数或浮点运算次数．在实际应用中，这两种定义复杂度的方式均很常见．如果能较准确地估计每次迭代的运算量，则可以由算法所需迭代次数推出所需浮点运算次数．我们用具体的例子来进一步解释算法复杂度．设某一算法产生的迭代序列 $\{x^k\}$ 满足

$$f(x^k)-f(x^*)\leqslant\frac{c}{\sqrt{k}},\quad\forall\,k>0,$$

其中 $c>0$ 为常数，x^* 为全局极小点．如果需要计算算法满足精度 $f(x^k)-f(x^*)\leqslant\varepsilon$ 所需的迭代次数，只需令 $\frac{c}{\sqrt{k}}\leqslant\varepsilon$ 则得到 $k\geqslant\frac{c^2}{\varepsilon^2}$，因此该优化算法对应的（迭代次数）复杂度为 $N(\varepsilon)=\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$．注意，渐进收敛速度更多的是考虑迭代次数充分大的情形，而复杂度给出了算法迭代有限步之后产生的解与最优解之间的定量关系，因此近年来受到人们广泛关注．

1.5 总结

本章简要介绍了优化问题的应用背景、一般形式以及一些基本概念．对于优化问题的更多分类、优化领域关心的热点问题，我们会在三章中进一步介绍．对于优化算法的收敛准则、收敛性以及收敛速度，我们会在介绍算法

的时候再具体展开. 本书也会围绕上面介绍的优化算法的四个设计技巧，针对不同类别的问题，来具体地展示相应的算法构造以及有效性分析.

习题 1

1.1 考虑稀疏优化问题，我们已经直观地讨论了在 ℓ_0，ℓ_1，ℓ_2 三种范数下问题的解的可能形式. 针对一般的 ℓ_p “范数”：

$$\|x\|_p \xlongequal{\text{def}} (\sum_{i=1}^{n}|x|^p)^{1/p}, \quad 0<p<2,$$

我们考虑优化问题：

$$\begin{aligned} \min \quad & \|x\|_p, \\ \text{s.t.} \quad & Ax=b. \end{aligned}$$

试着用几何直观的方式（类似于图 1.2）来说明当 $p\in(0,2)$ 取何值时，该优化问题的解可能具有稀疏性.

1.2 给定一个函数 $f(x):\mathbb{R}^n\to\mathbb{R}$ 及其一个局部最优点 x^*，则该点沿任何方向 $d\in\mathbb{R}^n$ 也是局部最优的，即 0 为函数 $\phi(\alpha)\xlongequal{\text{def}} f(x^*+\alpha d)$ 的一个局部最优解. 反之，如果 x^* 沿任何方向 $d\in\mathbb{R}^n$ 都是局部最优解，则 x^* 是否为 $f(x)$ 的一个局部最优解？若是，请给出证明；若不是，请给出反例.

1.3 考虑函数 $f(x)=x_1^2+x_2^2$，$x=(x_1,x_2)\in\mathbb{R}^2$，以及迭代点列 $x^k=(1+\frac{1}{2^k})(\cos k,\sin k)^{\mathrm{T}}, k=1,2,\cdots$，请说明

(a) $\{f(x^{k+1})\}$ 是否收敛？若收敛，给出 Q-收敛速度；

(b) $\{x^{k+1}\}$ 是否收敛？若收敛，给出 Q-收敛速度.

第二章　基础知识

在介绍具体的最优化模型、理论和算法之前，我们先介绍一些必备的基础知识. 本章中从范数和导数讲起，接着介绍广义实值函数、凸集、凸函数、共轭函数和次梯度等凸分析方面的重要概念和相关结论. 这一章的部分内容可能在较后的章节中才会用到，读者阅读时可按需选择，如共轭函数、次梯度可在学习相关优化算法时再阅读.

2.1　范数

和标量不同，我们不能简单地按照元素大小来比较不同的向量和矩阵. 向量范数和矩阵范数给出了一种长度计量方式. 我们首先介绍向量范数.

2.1.1　向量范数

定义 2.1 (范数)　称一个从向量空间 $\mathbb{R}^n$ 到实数域 $\mathbb{R}$ 的非负函数 $\|\cdot\|$ 为**范数**，如果它满足：

(1) 正定性：对于所有的 $v \in \mathbb{R}^n$，有 $\|v\| \geqslant 0$，且 $\|v\| = 0$ 当且仅当 $v = 0$；

(2) 齐次性：对于所有的 $v \in \mathbb{R}^n$ 和 $\alpha \in \mathbb{R}$，有 $\|\alpha v\| = |\alpha|\|v\|$；

(3) 三角不等式：对于所有的 v，$w \in \mathbb{R}^n$，有 $\|v + w\| \leqslant \|v\| + \|w\|$.

最常用的向量范数为 ℓ_p 范数 $(p \geqslant 1)$：

$$\|v\|_p = (|v_1|^p + |v_2|^p + \cdots + |v_n|^p)^{\frac{1}{p}};$$

当 $p = \infty$ 时，ℓ_∞ 范数定义为

$$\|v\|_\infty = \max_i |v_i|.$$

其中 $p=1,2,\infty$ 的情形最重要，分别记为 $\|\cdot\|_1, \|\cdot\|_2$ 和 $\|\cdot\|_\infty$. 在不引起歧义的情况下，我们有时省略 ℓ_2 范数的角标，记为 $\|\cdot\|$. 在最优化问题算法构造和分析中，也常常遇到由正定矩阵 A 诱导的范数，即 $\|x\|_A \xlongequal{\text{def}} \sqrt{x^{\mathrm{T}}Ax}$. 根据正定矩阵的定义，很容易验证 $\|\cdot\|_A$ 定义了一个范数.

对向量的 ℓ_2 范数，我们有常用的柯西（Cauchy）不等式：

命题 2.1 (柯西不等式) *设 $a,b\in\mathbb{R}^n$，则*

$$|a^{\mathrm{T}}b| \leqslant \|a\|_2\|b\|_2,$$

等号成立当且仅当 a 与 b 线性相关.

2.1.2 矩阵范数

和向量范数类似，矩阵范数是定义在矩阵空间上的非负函数，并且满足正定性、齐次性和三角不等式. 向量的 ℓ_p 范数可以比较容易地推广到矩阵的 ℓ_p 范数，本书常用 $p=1,2$ 的情形. 当 $p=1$ 时，矩阵 $A\in\mathbb{R}^{m\times n}$ 的 ℓ_1 范数定义为

$$\|A\|_1 = \sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|,$$

即 $\|A\|_1$ 为 A 中所有元素绝对值的和. 当 $p=2$ 时，此时得到的是矩阵的 Frobenius 范数（下称 F 范数），记为 $\|A\|_F$. 它可以看成是向量的 ℓ_2 范数的推广，即所有元素平方和开根号：

$$\|A\|_F = \sqrt{\mathrm{Tr}(AA^{\mathrm{T}})} = \sqrt{\sum_{i,j}a_{ij}^2}. \tag{2.1.1}$$

这里，$\mathrm{Tr}(X)$ 表示方阵 X 的迹. 矩阵的 F 范数具有正交不变性，即对于任意的正交矩阵 $U\in\mathbb{R}^{m\times m}, V\in\mathbb{R}^{n\times n}$, 我们有

$$\begin{aligned}\|UAV\|_F^2 &= \mathrm{Tr}(UAVV^{\mathrm{T}}A^{\mathrm{T}}U^{\mathrm{T}}) = \mathrm{Tr}(UAA^{\mathrm{T}}U^{\mathrm{T}})\\ &= \mathrm{Tr}(AA^{\mathrm{T}}U^{\mathrm{T}}U) = \mathrm{Tr}(AA^{\mathrm{T}}) = \|A\|_F^2,\end{aligned}$$

其中第三个等号成立是因为 $\mathrm{Tr}(AB)=\mathrm{Tr}(BA)$.

除了从向量范数直接推广以外，矩阵范数还可以由向量范数诱导出来，一般称这种范数为**算子范数**. 给定矩阵 $A\in\mathbb{R}^{m\times n}$，以及 m 维和 n 维空间的向量范数 $\|\cdot\|_{(m)}$ 和 $\|\cdot\|_{(n)}$，其诱导的矩阵范数定义如下：

$$\|A\|_{(m,n)} = \max_{x\in\mathbb{R}^n,\|x\|_{(n)}=1}\|Ax\|_{(m)},$$

容易验证 $\|\cdot\|_{(m,n)}$ 满足范数的定义. 如果将 $\|\cdot\|_{(m)}$ 和 $\|\cdot\|_{(n)}$ 都取为相应向量空间的 ℓ_p 范数，我们可以得到矩阵的 p 范数. 本书经常用到的是矩阵的 2 范数，即

$$\|A\|_2 = \max_{x\in\mathbb{R}^n, \|x\|_2=1} \|Ax\|_2.$$

容易验证（见习题2.1），矩阵的 2 范数是该矩阵的最大奇异值. 根据算子范数的定义，所有算子范数都满足如下性质：

$$\|Ax\|_{(m)} \leqslant \|A\|_{(m,n)}\|x\|_{(n)}. \tag{2.1.2}$$

例如当 $m=n=2$ 时，$\|Ax\|_2 \leqslant \|A\|_2\|x\|_2$. 性质(2.1.2)又被称为矩阵范数的**相容性**，即 $\|\cdot\|_{(m,n)}$ 与 $\|\cdot\|_{(m)}$ 和 $\|\cdot\|_{(n)}$ 是相容的. 并非所有矩阵范数都与给定的向量范数相容，在今后的应用中读者需要注意这一问题.

注 2.1 和矩阵 2 范数类似，向量的 ℓ_1 范数以及 ℓ_∞ 范数均可诱导出相应的矩阵范数（分别为矩阵的 1 范数和无穷范数），在多数数值代数教材中将它们记为 $\|\cdot\|_1$ 和 $\|\cdot\|_\infty$. 然而本书较少涉及这两个范数，因此我们将 $\|A\|_1$ 定义为矩阵 A 中所有元素绝对值的和. 读者应当注意它和其他数值代数教材中定义的不同.

除了矩阵 2 范数以外，另一个常用的矩阵范数为**核范数**. 给定矩阵 $A\in\mathbb{R}^{m\times n}$，其核范数定义为

$$\|A\|_* = \sum_{i=1}^{r}\sigma_i,$$

其中 $\sigma_i, i=1,2,\cdots,r$ 为 A 的所有非零奇异值，$r=\mathrm{rank}(A)$. 类似于向量的 ℓ_1 范数的保稀疏性，我们也经常通过限制矩阵的核范数来保证矩阵的低秩性. 同时，根据范数的三角不等式（下文中的凸性），相应的优化问题可以有效求解.

2.1.3 矩阵内积

对于矩阵空间 $\mathbb{R}^{m\times n}$ 的两个矩阵 A 和 B，除了定义它们各自的范数以外，我们还可以定义它们之间的内积. 范数一般用来衡量矩阵的模的大小，而内积一般用来表征两个矩阵（或其张成的空间）之间的夹角. 这里，我们介绍一种常用的内积——Frobenius 内积. $m\times n$ 矩阵 A 和 B 的 Frobenius 内积定义为

$$\langle A,B\rangle \xlongequal{\text{def}} \mathrm{Tr}(AB^{\mathrm{T}}) = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}b_{ij}.$$

易知其为两个矩阵逐分量相乘的和，因而满足内积的定义. 当 $A = B$ 时，$\langle A, B\rangle$ 等于矩阵 A 的 F 范数的平方.

和向量范数相似，我们也有矩阵范数对应的柯西不等式：

命题 2.2 (矩阵范数的柯西不等式) 设 $A, B \in \mathbb{R}^{m\times n}$，则

$$|\langle A, B\rangle| \leqslant \|A\|_F \|B\|_F,$$

等号成立当且仅当 A 和 B 线性相关.

2.2 导数

为了分析可微最优化问题的性质，我们需要知道目标函数和约束函数的导数信息. 在算法设计中，当优化问题没有显式解时，我们也往往通过函数值和导数信息来构造容易求解的子问题. 利用目标函数和约束函数的导数信息，可以确保构造的子问题具有很好的逼近性质，从而构造各种各样有效的算法. 本节将介绍有关导数的内容.

2.2.1 梯度与海瑟矩阵

在数学分析课程中，我们已经学过多元函数微分学. 这一小节，我们首先回顾梯度和海瑟（Hessian）矩阵的定义，之后介绍多元可微函数的一些重要性质.

定义 2.2 (梯度) 给定函数 $f : \mathbb{R}^n \to \mathbb{R}$，且 f 在点 x 的一个邻域内有意义，若存在向量 $g \in \mathbb{R}^n$ 满足

$$\lim_{p\to 0} \frac{f(x+p) - f(x) - g^{\mathrm{T}} p}{\|p\|} = 0, \tag{2.2.1}$$

其中 $\|\cdot\|$ 是任意的向量范数，就称 f 在点 x 处**可微**（或 Fréchet 可微）. 此时 g 称为 f 在点 x 处的**梯度**，记作 $\nabla f(x)$. 如果对区域 D 上的每一个点 x 都有 $\nabla f(x)$ 存在，则称 f 在 D 上可微.

若 f 在点 x 处的梯度存在，在 (2.2.1) 式中令 $p = \varepsilon e_i$，e_i 是第 i 个分量为 1 的单位向量，可知 $\nabla f(x)$ 的第 i 个分量为 $\dfrac{\partial f(x)}{\partial x_i}$. 因此，

$$\nabla f(x) = \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \cdots, \frac{\partial f(x)}{\partial x_n}\right]^{\mathrm{T}}.$$

如果只关心对一部分变量的梯度，可以通过对 ∇ 加下标来表示. 例如，$\nabla_x f(x,y)$ 表示将 y 视为常数时 f 关于 x 的梯度.

对应于一元函数的二阶导数，对于多元函数我们可以定义其海瑟矩阵.

定义 2.3 (海瑟矩阵) *如果函数 $f(x):\mathbb{R}^n \to \mathbb{R}$ 在点 x 处的二阶偏导数 $\dfrac{\partial^2 f(x)}{\partial x_i \partial x_j}$ $i,j=1,2,\cdots,n$ 都存在，则*

$$\nabla^2 f(x) = \begin{bmatrix} \dfrac{\partial^2 f(x)}{\partial x_1^2} & \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_3} & \cdots & \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f(x)}{\partial x_2^2} & \dfrac{\partial^2 f(x)}{\partial x_2 \partial x_3} & \cdots & \dfrac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \vdots & & \vdots \\ \dfrac{\partial^2 f(x)}{\partial x_n \partial x_1} & \dfrac{\partial^2 f(x)}{\partial x_n \partial x_2} & \dfrac{\partial^2 f(x)}{\partial x_n \partial x_3} & \cdots & \dfrac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}$$

称为 f 在点 x 处的海瑟矩阵.

当 $\nabla^2 f(x)$ 在区域 D 上的每个点 x 处都存在时，称 f 在 D 上二阶可微. 若 $\nabla^2 f(x)$ 在 D 上还连续，则称 f 在 D 上二阶连续可微，可以证明此时海瑟矩阵是一个对称矩阵.

当 $f:\mathbb{R}^n \to \mathbb{R}^m$ 是向量值函数时，我们可以定义它的雅可比 (Jacobi) 矩阵 $J(x) \in \mathbb{R}^{m\times n}$，它的第 i 行是分量 $f_i(x)$ 梯度的转置，即

$$J(x) = \begin{bmatrix} \dfrac{\partial f_1(x)}{\partial x_1} & \dfrac{\partial f_1(x)}{\partial x_2} & \cdots & \dfrac{\partial f_1(x)}{\partial x_n} \\ \dfrac{\partial f_2(x)}{\partial x_1} & \dfrac{\partial f_2(x)}{\partial x_2} & \cdots & \dfrac{\partial f_2(x)}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \dfrac{\partial f_m(x)}{\partial x_1} & \dfrac{\partial f_m(x)}{\partial x_2} & \cdots & \dfrac{\partial f_m(x)}{\partial x_n} \end{bmatrix}.$$

此外容易看出，梯度 $\nabla f(x)$ 的雅可比矩阵就是 $f(x)$ 的海瑟矩阵.

类似于一元函数的泰勒展开，对于多元函数，我们不加证明地给出如下形式的泰勒展开：

定理 2.1 *设 $f:\mathbb{R}^n \to \mathbb{R}$ 是连续可微的，$p \in \mathbb{R}^n$ 为向量，那么*

$$f(x+p) = f(x) + \nabla f(x+tp)^{\mathrm{T}} p,$$

其中 $0<t<1$. 进一步地，如果 f 是二阶连续可微的，则

$$\nabla f(x+p)=\nabla f(x)+\int_0^1 \nabla^2 f(x+tp)p\mathrm{d}t,$$
$$f(x+p)=f(x)+\nabla f(x)^{\mathrm{T}}p+\frac{1}{2}p^{\mathrm{T}}\nabla^2 f(x+tp)p,$$

其中 $0<t<1$.

在这一小节的最后，我们介绍一类特殊的可微函数——梯度利普希茨 (Lipschitz) 连续的函数. 该类函数在很多优化算法收敛性证明中起着关键作用.

定义 2.4 (梯度利普希茨连续) 给定可微函数 f，若存在 $L>0$，对任意的 $x,y\in\mathbf{dom}\,f$ 有

$$\|\nabla f(x)-\nabla f(y)\|\leqslant L\|x-y\|, \tag{2.2.2}$$

则称 f 是**梯度利普希茨连续的**，相应利普希茨常数为 L. 有时也简记为**梯度 L-利普希茨连续**或 **L-光滑**.

梯度利普希茨连续表明 $\nabla f(x)$ 的变化可以被自变量 x 的变化所控制，满足该性质的函数具有很多好的性质，一个重要的性质是其具有**二次上界**.

引理 2.1 (二次上界) 设可微函数 $f(x)$ 的定义域 $\mathbf{dom}\,f=\mathbb{R}^n$，且为梯度 L-利普希茨连续的，则函数 $f(x)$ 有二次上界：

$$f(y)\leqslant f(x)+\nabla f(x)^{\mathrm{T}}(y-x)+\frac{L}{2}\|y-x\|^2,\quad \forall\, x,y\in\mathbf{dom}\,f. \tag{2.2.3}$$

证明. 对任意的 $x,y\in\mathbb{R}^n$，构造辅助函数

$$g(t)=f(x+t(y-x)),\quad t\in[0,1]. \tag{2.2.4}$$

显然 $g(0)=f(x)$，$g(1)=f(y)$，以及

$$g'(t)=\nabla f(x+t(y-x))^{\mathrm{T}}(y-x).$$

由等式

$$g(1)-g(0)=\int_0^1 g'(t)\mathrm{d}t$$

可知

$$\begin{aligned}
&f(y)-f(x)-\nabla f(x)^{\mathrm{T}}(y-x)\\
=&\int_0^1 (g'(t)-g'(0))\mathrm{d}t\\
=&\int_0^1 (\nabla f(x+t(y-x))-\nabla f(x))^{\mathrm{T}}(y-x)\mathrm{d}t\\
\leqslant&\int_0^1 \|\nabla f(x+t(y-x))-\nabla f(x)\|\|y-x\|\mathrm{d}t\\
\leqslant&\int_0^1 L\|y-x\|^2 t\mathrm{d}t=\frac{L}{2}\|y-x\|^2,
\end{aligned}$$

其中最后一行的不等式利用了梯度利普希茨连续的条件 (2.2.2). 整理可得 (2.2.3) 式成立. □

引理 2.1 实际上指的是 $f(x)$ 可被一个二次函数上界所控制，即要求 $f(x)$ 的增长速度不超过二次. 实际上，该引理对 $f(x)$ 定义域的要求可减弱为 $\mathbf{dom}\, f$ 是凸集（见定义 2.13），此条件的作用是保证证明中的 $g(t)$ 当 $t\in[0,1]$ 时是有定义的.

若 f 是梯度利普希茨连续的，且有一个全局极小点 x^*，一个重要的推论就是我们能够利用二次上界(2.2.3) 来估计 $f(x)-f(x^*)$ 的大小，其中 x 可以是定义域中的任意一点.

推论 2.1 *设可微函数 $f(x)$ 的定义域为 $\mathbb{R}^n$ 且存在一个全局极小点 x^*，若 $f(x)$ 为梯度 L-利普希茨连续的，则对任意的 x 有*

$$\frac{1}{2L}\|\nabla f(x)\|^2\leqslant f(x)-f(x^*). \tag{2.2.5}$$

证明. 由于 x^* 是全局极小点，应用二次上界(2.2.3)有

$$f(x^*)\leqslant f(y)\leqslant f(x)+\nabla f(x)^{\mathrm{T}}(y-x)+\frac{L}{2}\|y-x\|^2.$$

在这里固定 x，注意到上式对于任意的 y 均成立，因此可对上式不等号右边取下确界：

$$\begin{aligned}
f(x^*)&\leqslant \inf_{y\in\mathbb{R}^n}\left\{f(x)+\nabla f(x)^{\mathrm{T}}(y-x)+\frac{L}{2}\|y-x\|^2\right\}\\
&=f(x)-\frac{1}{2L}\|\nabla f(x)\|^2.
\end{aligned}$$

□

推论 2.1 证明的最后一步应用了二次函数的性质：当 $y=x-\dfrac{\nabla f(x)}{L}$ 时取到最小值. 有关二次函数最优性条件将在第四章中进一步讨论.

2.2.2 矩阵变量函数的导数

多元函数梯度的定义可以推广到变量是矩阵的情形. 对于以 $m \times n$ 矩阵 X 为自变量的函数 $f(X)$，若存在矩阵 $G \in \mathbb{R}^{m\times n}$ 满足

$$\lim_{V\to 0} \frac{f(X+V) - f(X) - \langle G, V\rangle}{\|V\|} = 0,$$

其中 $\|\cdot\|$ 是任意矩阵范数，就称矩阵变量函数 f 在 X 处 Fréchet 可微，称 G 为 f 在 Fréchet 可微意义下的梯度. 类似于向量情形，矩阵变量函数 $f(X)$ 的梯度可以用其偏导数表示为

$$\nabla f(x) = \begin{bmatrix} \dfrac{\partial f}{\partial x_{11}} & \dfrac{\partial f}{\partial x_{12}} & \cdots & \dfrac{\partial f}{\partial x_{1n}} \\ \dfrac{\partial f}{\partial x_{21}} & \dfrac{\partial f}{\partial x_{22}} & \cdots & \dfrac{\partial f}{\partial x_{2n}} \\ \vdots & \vdots & & \vdots \\ \dfrac{\partial f}{\partial x_{m1}} & \dfrac{\partial f}{\partial x_{m2}} & \cdots & \dfrac{\partial f}{\partial x_{mn}} \end{bmatrix}.$$

其中 $\dfrac{\partial f}{\partial x_{ij}}$ 表示 f 关于 x_{ij} 的偏导数.

在实际应用中，矩阵 Fréchet 可微的定义和使用往往比较繁琐，为此我们需要介绍另一种定义——Gâteaux 可微.

定义 2.5 (Gâteaux 可微) 设 $f(X)$ 为矩阵变量函数，如果存在矩阵 $G \in \mathbb{R}^{m\times n}$，对任意方向 $V \in \mathbb{R}^{m\times n}$ 满足

$$\lim_{t\to 0} \frac{f(X+tV) - f(X) - t\langle G, V\rangle}{t} = 0, \tag{2.2.6}$$

则称 f 关于 X 是 Gâteaux 可微的. 满足(2.2.6)式的 G 称为 f 在 X 处在 Gâteaux 可微意义下的梯度.

可以证明，当 f 是 Fréchet 可微函数时，f 也是 Gâteaux 可微的，且这两种意义下的梯度相等.

在实际中，由于 Gâteaux 可微定义式更容易操作，因此通常是利用 (2.2.6) 式进行矩阵变量函数 $f(X)$ 的求导运算. 我们以下面的例子来具体说明.

例 2.1

(1) 考虑线性函数：$f(X)=\mathrm{Tr}(AX^{\mathrm{T}}B)$，其中 $A\in\mathbb{R}^{p\times n}, B\in\mathbb{R}^{m\times p}, X\in\mathbb{R}^{m\times n}$，对任意方向 $V\in\mathbb{R}^{m\times n}$ 以及 $t\in\mathbb{R}$，有

$$\begin{aligned}\lim_{t\to 0}\frac{f(X+tV)-f(X)}{t}&=\lim_{t\to 0}\frac{\mathrm{Tr}(A(X+tV)^{\mathrm{T}}B)-\mathrm{Tr}(AX^{\mathrm{T}}B)}{t}\\&=\mathrm{Tr}(AV^{\mathrm{T}}B)=\langle BA,V\rangle.\end{aligned}$$

因此，$\nabla f(X)=BA$.

(2) 考虑二次函数：$f(X,Y)=\frac{1}{2}\|XY-A\|_F^2$，其中 $(X,Y)\in\mathbb{R}^{m\times p}\times\mathbb{R}^{p\times n}$，对变量 Y，其中 $X\in\mathbb{R}^{m\times p}$，$Y\in\mathbb{R}^{p\times n}$，取任意方向 V 以及充分小的 $t\in\mathbb{R}$，有

$$\begin{aligned}f(X,Y+tV)-f(X,Y)&=\frac{1}{2}\|X(Y+tV)-A\|_F^2-\frac{1}{2}\|XY-A\|_F^2\\&=\langle tXV,XY-A\rangle+\frac{1}{2}t^2\|XV\|_F^2\\&=t\left\langle V,X^{\mathrm{T}}(XY-A)\right\rangle+\mathcal{O}(t^2).\end{aligned}$$

由定义可知 $\dfrac{\partial f}{\partial Y}=X^{\mathrm{T}}(XY-A)$.

对变量 X，取任意方向 V 以及充分小的 $t\in\mathbb{R}$，有

$$\begin{aligned}f(X+tV,Y)-f(X,Y)&=\frac{1}{2}\|(X+tV)Y-A\|_F^2-\frac{1}{2}\|XY-A\|_F^2\\&=\langle tVY,XY-A\rangle+\frac{1}{2}t^2\|VY\|_F^2\\&=t\left\langle V,(XY-A)Y^{\mathrm{T}}\right\rangle+\mathcal{O}(t^2).\end{aligned}$$

由定义可知 $\dfrac{\partial f}{\partial X}=(XY-A)Y^{\mathrm{T}}$.

在对函数求导的过程中，应当注意函数的自变量和相应的导数应该有相同的维数. 例如自变量 $X\in\mathbb{R}^{m\times n}$，那么其矩阵导数 $\nabla f(X)\in\mathbb{R}^{m\times n}$. 这个要求在对矩阵变量函数求导时非常容易被忽略，检查一个矩阵变量函数的导数是否正确的第一步是要验证其维数是否和对应的自变量相符. 读者可利用例 2.1 的 (2) 来加深对矩阵变量函数导数的理解.

2.3 广义实值函数

在数学分析课程中我们学习了函数的基本概念：函数是从向量空间 $\mathbb{R}^n$ 到实数域 $\mathbb{R}$ 的映射. 而在最优化领域，经常涉及对某个函数其中的一个变

量取 inf（sup）操作，这导致函数的取值可能为无穷. 为了能够更方便地描述优化问题，我们需要对函数的定义进行某种扩展.

定义 2.6 (广义实值函数) 令 $\overline{\mathbb{R}} \xlongequal{\text{def}} \mathbb{R} \cup \{\pm\infty\}$ 为广义实数空间，则映射 $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ 称为**广义实值函数**.

从广义实值函数的定义可以看出，其值域多了两个特殊的值 $\pm\infty$. 和数学分析一样，我们规定

$$-\infty < a < +\infty, \quad \forall\, a \in \mathbb{R}$$

以及

$$(+\infty) + (+\infty) = +\infty, \quad +\infty + a = +\infty, \forall\, a \in \mathbb{R}.$$

2.3.1 适当函数

适当函数是一类很重要的广义实值函数，很多最优化理论都是建立在适当函数之上的.

定义 2.7 (适当函数) 给定广义实值函数 f 和非空集合 $\mathcal{X}$. 如果存在 $x \in \mathcal{X}$ 使得 $f(x) < +\infty$，并且对任意的 $x \in \mathcal{X}$，都有 $f(x) > -\infty$，那么称函数 f 关于集合 $\mathcal{X}$ 是适当的.

概括来说，适当函数 f 的特点是 “至少有一处取值不为正无穷”，以及 “处处取值不为负无穷”. 对最优化问题 $\min\limits_{x} f(x)$，适当函数可以帮助去掉一些我们不感兴趣的函数，从而在一个比较合理的函数类中考虑最优化问题. 我们约定：**在本书中若无特殊说明，定理中所讨论的函数均为适当函数.**

对于适当函数 f，规定其定义域

$$\mathbf{dom}\, f = \{x \mid f(x) < +\infty\}.$$

正是因为适当函数的最小值不可能在函数值为无穷处取到，因此 $\mathbf{dom}\, f$ 的定义方式是自然的.

2.3.2 闭函数

闭函数是另一类重要的广义实值函数，本书后面章节的许多定理都建立在闭函数之上. 在数学分析课程中我们接触过连续函数，本小节介绍的闭函数可以看成是连续函数的一种推广.

在介绍闭函数之前，我们先引入一些基本概念.

1. 下水平集

下水平集是描述实值函数取值情况的一个重要概念. 为此有如下定义:

定义 2.8 (α-下水平集) 对于广义实值函数 $f\colon \mathbb{R}^n \to \overline{\mathbb{R}}$,

$$C_\alpha = \{x \,|\, f(x) \leqslant \alpha\}$$

称为 f 的**α-下水平集**.

在最优化问题中, 多数情况都要对函数 $f(x)$ 求极小值, 通过研究 α-下水平集可以知道具体在哪些点处 $f(x)$ 的值不超过 α. 若 C_α 非空, 我们知道 $f(x)$ 的全局极小点 (若存在) 一定落在 C_α 中, 因此也就无需考虑 C_α 之外的点.

2. 上方图

上方图是从集合的角度来描述一个函数的具体性质. 我们有如下定义:

定义 2.9 (上方图) 对于广义实值函数 $f\colon \mathbb{R}^n \to \overline{\mathbb{R}}$,

$$\mathbf{epi}\, f = \{(x,t) \in \mathbb{R}^{n+1} \,|\, f(x) \leqslant t\}$$

称为 f 的**上方图**.

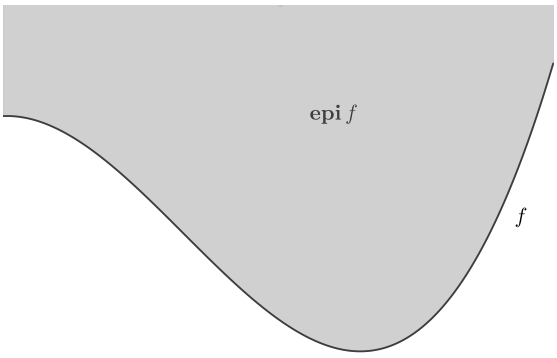


图 2.1 函数 f 和其上方图 $\mathbf{epi}\, f$

上方图的一个直观的例子如图 2.1 所示. 上方图将函数和集合建立了联系, f 的很多性质都可以在 $\mathbf{epi}\, f$ 上得到体现. 在后面的分析中将看到, 我们可以通过 $\mathbf{epi}\, f$ 的一些性质来反推 f 的性质.

3. 闭函数下半连续函数

基于前面介绍的一些基本概念，我们可以给出闭函数和下半连续函数的定义.

定义 2.10 (闭函数) 设 $f\colon \mathbb{R}^n \to \overline{\mathbb{R}}$ 为广义实值函数，若 $\mathbf{epi} f$ 为闭集，则称 f 为**闭函数**.

定义 2.11 (下半连续函数) 设广义实值函数 $f\colon \mathbb{R}^n \to \overline{\mathbb{R}}$，若对任意的 $x \in \mathbb{R}^n$，有

$$\liminf_{y \to x} f(y) \geqslant f(x),$$

则 $f(x)$ 为**下半连续函数**.

如图2.2所示，$f(x)$ 为 $\mathbb{R}$ 上的下半连续函数.

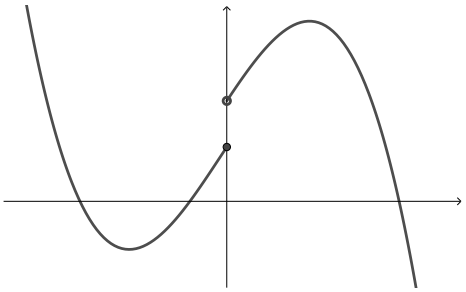


图 2.2 下半连续函数 $f(x)$

有趣的是，虽然表面上看这两种函数的定义方式截然不同，但闭函数和下半连续函数是等价的. 我们不加证明地给出如下定理：

定理 2.2 (闭函数和下半连续函数的等价性 [5]) 设广义实值函数 $f\colon \mathbb{R}^n \to \overline{\mathbb{R}}$，则以下命题等价：

(1) $f(x)$ 的任意 α-下水平集都是闭集；

(2) $f(x)$ 是下半连续的；

(3) $f(x)$ 是闭函数.

以上等价性为我们之后证明定理提供了很大的方便. 由于下半连续函数也具有某种连续性，第四章也会介绍其相应的最小值存在定理. 在许多文献中闭函数和下半连续函数往往只出现一种定义，读者应当注意这个等价关系.

2.4 凸集

2.4.1 凸集的相关定义

对于 $\mathbb{R}^n$ 中的两个点 $x_1 \neq x_2$，形如

$$y = \theta x_1 + (1-\theta)x_2$$

的点形成了过点 x_1 和 x_2 的**直线**. 当 $0 \leqslant \theta \leqslant 1$ 时，这样的点形成了连接点 x_1 与 x_2 的**线段**.

定义 2.12 如果过集合 C 中任意两点的直线都在 C 内，则称 C 为**仿射集**，即

$$x_1, x_2 \in C \Longrightarrow \theta x_1 + (1-\theta)x_2 \in C, \forall \theta \in \mathbb{R}.$$

线性方程组 $Ax = b$ 的解集是仿射集. 反之，任何仿射集都可以表示成一个线性方程组的解集，读者可以自行验证.

定义 2.13 如果连接集合 C 中任意两点的线段都在 C 内，则称 C 为**凸集**，即

$$x_1, x_2 \in C \Longrightarrow \theta x_1 + (1-\theta)x_2 \in C, \forall 0 \leqslant \theta \leqslant 1.$$

从仿射集的定义容易看出仿射集都是凸集. 下面给出一些凸集和非凸集的例子.

例 2.2 在图 2.3 中，(a) 为凸集，(b)(c) 为非凸集，其中 (c) 不含部分边界点

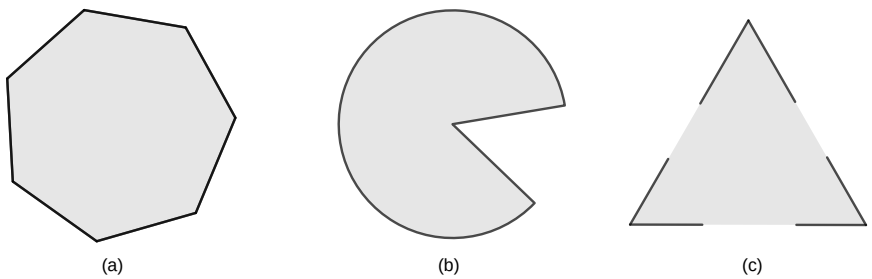


图 2.3 一个凸集和两个非凸集

从凸集可以引出凸组合和凸包等概念. 形如

$$x = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_k x_k,$$
$$1 = \theta_1 + \theta_2 + \cdots + \theta_k, \quad \theta_i \geqslant 0, i = 1,2,\cdots,k$$

的点称为 $x_1, x_2, \cdots, x_k$ 的**凸组合**. 集合 S 中点所有可能的凸组合构成的集合称作 S 的**凸包**，记作 $\mathbf{conv}\, S$. 实际上，$\mathbf{conv}\, S$ 是包含 S 的最小的凸集. 如图2.4所示，左边的为离散点集的凸包，右边的为扇形的凸包.

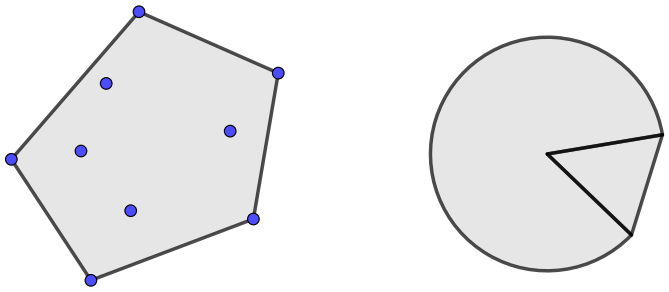


图 2.4 离散点集和扇形的凸包

若在凸组合的定义中去掉 $\theta_i \geqslant 0$ 的限制，我们可以得到**仿射包**的概念.

定义 2.14 (仿射包) 设 S 为 $\mathbb{R}^n$ 的子集，称如下集合为 S 的**仿射包**:

$$\{x \mid x = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_k x_k, \quad x_1, x_2, \cdots, x_k \in S, \quad \theta_1 + \theta_2 + \cdots + \theta_k = 1\},$$

记为 $\mathbf{affine}\, S$.

图 2.5 展示了 $\mathbb{R}^3$ 中圆盘 S 的仿射包，其为一个平面.

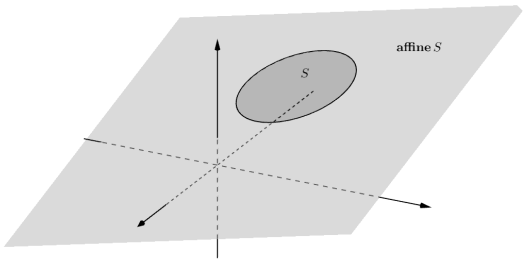


图 2.5 集合 S 的仿射包

一般而言，一个集合的仿射包实际上是包含该集合的最小的仿射集，这个概念在之后我们讨论凸问题最优性条件的时候会用到.

形如

$$x = \theta_1 x_1 + \theta_2 x_2, \quad \theta_1 > 0, \theta_2 > 0$$

的点称为点 x_1, x_2 的**锥组合**. 若集合 S 中任意点的锥组合都在 S 中，则称 S 为**凸锥**，如图2.6所示.

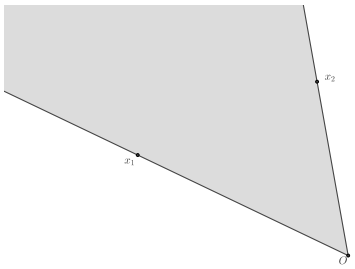


图 2.6 凸锥

2.4.2 重要的凸集

下面将介绍一些重要的凸集. 这些凸集在实际问题中常常会遇到.

1. 超平面和半空间

任取非零向量 a，形如 $\{x \mid a^{\mathrm{T}}x = b\}$ 的集合称为**超平面**，形如 $\{x \mid a^{\mathrm{T}}x \leqslant b\}$ 的集合称为**半空间**（如图2.7）．a 是对应的超平面和半空间的法向量．一个超平面将 $\mathbb{R}^n$ 分为两个半空间．容易看出，超平面是仿射集和凸集，半空间是凸集但不是仿射集.

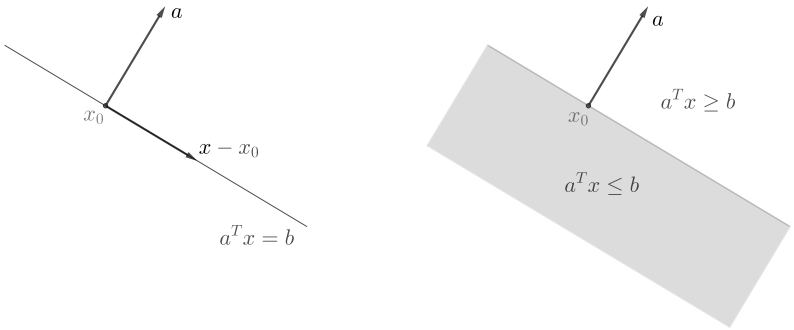


图 2.7 超平面和半空间

2. 球、椭球、锥

球和椭球也是常见的凸集．球是空间中到某个点距离（或两者差的范数）小于某个常数的点的集合，并将

$$B(x_c, r) = \{x \mid \|x - x_c\|_2 \leqslant r\} = \{x_c + ru \mid \|u\|_2 \leqslant 1\}$$

称为中心为 x_c，半径为 r 的 **(欧几里得) 球**．而形如

$$\{x \mid (x - x_c)^{\mathrm{T}} P^{-1} (x - x_c) \leqslant 1\}$$

的集合称为**椭球**，其中 $P \in \mathcal{S}^n_{++}$ (即 P 对称正定)．椭球的另一种表示为 $\{x_c + Au \mid \|u\|_2 \leqslant 1\}$，$A$ 为非奇异的方阵.

在定义一个球时，并不一定要使用欧几里得空间的距离．对于一般的范数，同样可以定义“球”．令 $\|\cdot\|$ 是任意一个范数，

$$\{x \mid \|x - x_c\| \leqslant r\}$$

称为中心为 x_c，半径为 r 的**范数球**. 另外，我们称集合

$$\{(x,t) \mid \|x\| \leqslant t\}$$

为**范数锥**. 欧几里得范数锥也称为**二次锥**. 范数球和范数锥都是凸集.

3. 多面体

我们把满足线性等式和不等式组的点的集合称为**多面体**，即

$$\{x \mid Ax \leqslant b, \quad Cx = d\},$$

其中 $A \in \mathbb{R}^{m\times n}$, $C \in \mathbb{R}^{p\times n}$, $x \leqslant y$ 表示向量 x 的每个分量均小于等于 y 的对应分量. 多面体是有限个半空间和超平面的交集，因此是凸集.

4. （半）正定锥

记 $\mathcal{S}^n$ 为 $n \times n$ 对称矩阵的集合，$\mathcal{S}^n_+ = \{X \in \mathcal{S}^n \mid X \succeq 0\}$ 为 $n \times n$ 半正定矩阵的集合，$\mathcal{S}^n_{++} = \{X \in \mathcal{S}^n \mid X \succ 0\}$ 为 $n \times n$ 正定矩阵的集合. 容易证明 $\mathcal{S}^n_+$ 是凸锥，因此 $\mathcal{S}^n_+$ 又称为**半正定锥**. 图 2.8 展示了二维半正定锥的几何形状.

例 2.3 $\begin{bmatrix} x & y \\ y & z \end{bmatrix} \in \mathcal{S}^2_+$，点 (x,y,z) 构成的图形的边界如图 2.8 所示.

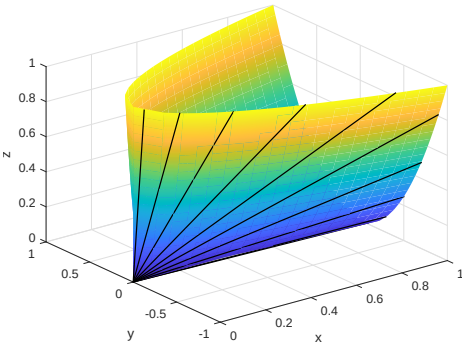


图 2.8 二维半正定锥 $\mathcal{S}^2_+$ 的边界

2.4.3 保凸的运算

下面介绍证明一个集合（设为 C）为凸集的两种方式. 第一种是利用定义

$$x_1, x_2 \in C, 0 \leqslant \theta \leqslant 1 \Longrightarrow \theta x_1 + (1-\theta)x_2 \in C$$

来证明集合 C 是凸集. 第二种方法是说明集合 C 可由简单的凸集 (超平面、半空间、范数球等) 经过保凸的运算后得到. 为此，我们需要掌握一些常见的保凸运算. 我们将不加证明地指出，**任意多个凸集的交以及仿射变换都是保凸的**.

注意到缩放、平移和投影变换都是仿射变换，因此凸集经过缩放、平移或投影的像仍是凸集. 利用仿射变换保凸的性质，可以证明线性矩阵不等式的解集 $\{x \mid x_1A_1 + x_2A_2 + \cdots + x_mA_m \preceq B\}$ 是凸集 $(A_i, i = 1,2,\cdots,m, B \in \mathcal{S}^p)$，双曲锥 $\{x \mid x^{\mathrm{T}}Px \leqslant (c^{\mathrm{T}}x)^2, c^{\mathrm{T}}x \geqslant 0\}$ $(P \in \mathcal{S}_+^n)$ 是凸集.

2.4.4 分离超平面定理

这里，我们介绍凸集的一个重要性质，即可以用超平面分离不相交的凸集. 最基本的结果是分离超平面定理和支撑超平面定理.

定理 2.3 (分离超平面定理 [112]定理 11.3) *如果 C 和 D 是不相交的两个凸集，则存在非零向量 a 和常数 b，使得*

$$a^{\mathrm{T}}x \leqslant b, \quad \forall x \in C, \quad \text{且 } a^{\mathrm{T}}x \geqslant b, \quad \forall x \in D,$$

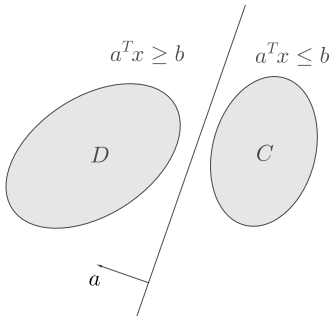


图 2.9 分离超平面

即超平面 $\{x \mid a^{\mathrm{T}}x = b\}$ 分离了 C 和 D (如图2.9).

严格分离 (即上式成立严格不等号) 需要更强的假设. 例如，当 C 是闭凸集，D 是单点集时，我们有如下严格分离定理.

定理 2.4 (严格分离定理 [20]例 2.20) *设 C 是闭凸集，点 $x_0 \notin C$，则存在非零向量 a 和常数 b，使得*

$$a^{\mathrm{T}}x < b, \forall x \in C, \quad \text{且} \quad a^{\mathrm{T}}x_0 > b.$$

上述严格分离定理要求点 $x_0 \notin C$. 当点 x_0 恰好在凸集 C 的边界上时，我们可以构造支撑超平面.

定义 2.15 (支撑超平面) 给定集合 C 及其边界上一点 x_0，如果 $a \neq 0$ 满足 $a^{\mathrm{T}}x \leqslant a^{\mathrm{T}}x_0, \forall x \in C$，那么称集合

$$\{x \,|\, a^{\mathrm{T}}x = a^{\mathrm{T}}x_0\}$$

为 C 在边界点 x_0 处的支撑超平面.

因此，点 x_0 和集合 C 也被该超平面分开. 从几何上来说，超平面 $\{x \,|\, a^{\mathrm{T}}x = a^{\mathrm{T}}x_0\}$ 与集合 C 在点 x_0 处相切并且半空间 $\{x \,|\, a^{\mathrm{T}}x \leqslant a^{\mathrm{T}}x_0\}$ 包含 C.

根据凸集的分离超平面定理，我们有如下支撑超平面定理.

定理 2.5 (支撑超平面定理 [112]推论 11.6.1) 如果 C 是凸集，则在 C 的任意边界点处都存在支撑超平面.

支撑超平面定理有非常强的几何直观：给定一个平面后，可把凸集边界上的任意一点当成支撑点将凸集放置在该平面上. 其他形状的集合一般没有这个性质，例如图 2.3 的 (b)，不可能以凹陷处为支撑点将其放置在水平面上.

2.5 凸函数

有了凸集的定义，我们来定义一类特殊的函数，即凸函数. 因在实际问题中的广泛应用，凸函数的研究得到了人们大量的关注.

2.5.1 凸函数的定义

定义 2.16 (凸函数) 设函数 f 为适当函数，如果 $\mathbf{dom}\, f$ 是凸集，且

$$f(\theta x + (1-\theta)y) \leqslant \theta f(x) + (1-\theta)f(y)$$

对所有 $x, y \in \mathbf{dom}\, f, 0 \leqslant \theta \leqslant 1$ 都成立，则称 f 是**凸函数**.

直观地来看，连接凸函数的图像上任意两点的线段都在函数图像上方（如图 2.10）.

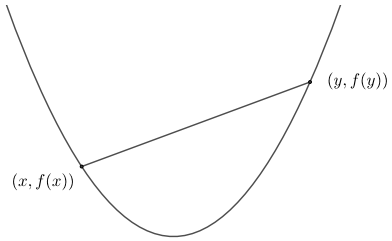


图 2.10 凸函数

相应地，我们也可以定义凹函数：若 $-f$ 是凸函数，则称 f 是**凹函数**. 只要改变一下符号，很多凸函数的性质都可以直接应用到凹函数上. 另外，如果 $\mathbf{dom}\, f$ 是凸集，且

$$f(\theta x + (1-\theta)y) < \theta f(x) + (1-\theta)f(y)$$

对所有的 $x,y \in \mathbf{dom}\, f$, $x \neq y$, $0 < \theta < 1$ 成立，则称 f 是**严格凸函数**. 除了严格凸函数以外，还有另一类常用的凸函数：**强凸函数**.

定义 2.17 (强凸函数) *若存在常数 $m > 0$，使得*

$$g(x) = f(x) - \frac{m}{2}\|x\|^2$$

*为凸函数，则称 $f(x)$ 为**强凸函数**，其中 m 为**强凸参数**. 为了方便我们也称 $f(x)$ 为 m-强凸函数.*

通过直接对 $g(x) = f(x) - \frac{m}{2}\|x\|^2$ 应用凸函数的定义，我们可得到另一个常用的强凸函数定义.

定义 2.18 (强凸函数的等价定义) *若存在常数 $m > 0$，使得对任意 $x,y \in \mathbf{dom}\, f$ 以及 $\theta \in (0,1)$，有*

$$f(\theta x + (1-\theta)y) \leqslant \theta f(x) + (1-\theta)f(y) - \frac{m}{2}\theta(1-\theta)\|x-y\|^2,$$

则称 $f(x)$ 为强凸函数，其中 m 为强凸参数.

强凸函数的两种定义侧重点不同：从定义2.17可以看出，强凸函数减去一个正定二次函数仍然是凸的；而从定义2.18可以看出，强凸函数一定是严格凸函数，当 $m = 0$ 时退化成凸函数. 无论从哪个定义出发，容易看出和凸

函数相比，强凸函数有更好的性质. 在后面很多算法的理论分析中，为了得到点列的收敛性以及更快的收敛速度，我们都要加上强凸这一条件.

此外，根据强凸函数的等价定义容易得出下面的结论：

命题 2.3 *设 f 为强凸函数且存在最小值，则 f 的最小值点唯一.*

证明. 采用反证法. 设 $x \neq y$ 均为 f 的最小值点，根据强凸函数的等价定义，取 $\theta \in (0,1)$，则有

$$\begin{aligned} f(\theta x + (1-\theta)y) &\leqslant \theta f(x) + (1-\theta) f(y) - \frac{m}{2}\theta(1-\theta)\|x-y\|^2 \\ &= f(x) - \frac{m}{2}\theta(1-\theta)\|x-y\|^2 \\ &< f(x), \end{aligned}$$

其中严格不等号成立是因为 $x \neq y$. 这显然和 $f(x)$ 为最小值矛盾，得证. □

注 2.2 *命题 2.3 中 f 存在最小值是前提. 强凸函数 f 的全局极小点不一定存在，例如 $f(x) = x^2$，* $\mathbf{dom}\, f = (1,2)$.

2.5.2 凸函数判定定理

凸函数的一个最基本的判定方式是：先将其限制在任意直线上，然后判断对应的一维函数是否是凸的. 如下面的定理所述，一个函数是凸函数当且仅当将函数限制在任意直线在定义域内的部分上时仍是凸的. 证明见 [20].

定理 2.6 *$f(x)$ 是凸函数当且仅当对任意的 $x \in \mathbf{dom}\, f, v \in \mathbb{R}^n$, $g : \mathbb{R} \to \mathbb{R}$,*

$$g(t) = f(x+tv), \quad \mathbf{dom}\ g = \{\, t \,|\, x + tv \in \mathbf{dom}\, f \,\}$$

是凸函数.

下面的例子说明如何利用定理 2.6 来判断一个函数是否为凸函数.

例 2.4 *$f(X) = -\ln\det X$ 是凸函数，其中 $\mathbf{dom}\, f = \mathcal{S}^n_{++}$.*

事实上，任取 $X \succ 0$ 以及方向 $V \in \mathcal{S}^n$，将 f 限制在直线 $X + tV$（t 满足 $X + tV \succ 0$）上，考虑函数 $g(t) = -\ln\det(X+tV)$. 那么

$$\begin{aligned} g(t) &= -\ln\det X - \ln\det(I + tX^{-1/2}VX^{-1/2}) \\ &= -\ln\det X - \sum_{i=1}^{n} \ln(1+t\lambda_i), \end{aligned}$$

其中 λ_i 是 $X^{-1/2}VX^{-1/2}$ 的第 i 个特征值. 对每个 $X \succ 0$ 以及方向 V, g 关于 t 是凸的. 因此 f 是凸的.

对于可微函数, 除了将其限制在直线上之外, 还可以利用其导数信息来判断它的凸性. 具体来说, 有如下的一阶条件:

定理 2.7 (一阶条件 [20]$^{\S 3.1.3}$) 对于定义在凸集上的可微函数 f, f 是凸函数当且仅当

$$f(y) \geqslant f(x) + \nabla f(x)^{\mathrm{T}}(y-x), \quad \forall x, y \in \mathbf{dom}\, f.$$

定理 2.7 说明可微凸函数 f 的图形始终在其任一点处切线的上方, 见图 2.11. 因此, 用可微凸函数 f 在任意一点处的一阶近似可以得到 f 的一个全局下界. 另一个常用的一阶条件是梯度单调性.

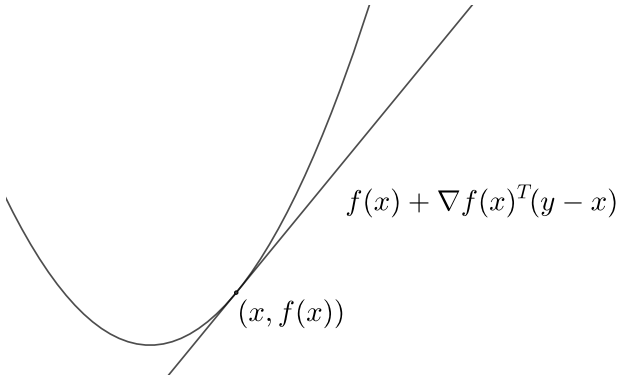


图 2.11 凸函数的全局下界

定理 2.8 (梯度单调性) 设 f 为可微函数, 则 f 为凸函数当且仅当 $\mathbf{dom}\, f$ 为凸集且 ∇f 为**单调映射**, 即

$$(\nabla f(x) - \nabla f(y))^{\mathrm{T}}(x-y) \geqslant 0, \quad \forall\, x, y \in \mathbf{dom}\, f.$$

证明. 先证必要性. 若 f 可微且为凸函数, 根据一阶条件, 我们有

$$f(y) \geqslant f(x) + \nabla f(x)^{\mathrm{T}}(y-x),$$
$$f(x) \geqslant f(y) + \nabla f(y)^{\mathrm{T}}(x-y).$$

将两式不等号左右两边相加即可得到结论.

再证充分性. 若 ∇f 为单调映射，构造一元辅助函数

$$g(t) = f(x + t(y - x)), \quad g'(t) = \nabla f(x + t(y - x))^{\mathrm{T}}(y - x)$$

由 ∇f 的单调性可知 $g'(t) \geqslant g'(0), \forall\, t \geqslant 0$. 因此

$$\begin{aligned} f(y) = g(1) &= g(0) + \int_0^1 g'(t)\mathrm{d}t \\ &\geqslant g(0) + g'(0) = f(x) + \nabla f(x)^{\mathrm{T}}(y - x). \end{aligned}$$

□

进一步地，如果函数二阶连续可微，我们可以得到下面的二阶条件：

定理 2.9 (二阶条件 [20]练习 3.8) *设 f 为定义在凸集上的二阶连续可微函数，则 f 是凸函数当且仅当*

$$\nabla^2 f(x) \succeq 0, \quad \forall x \in \mathbf{dom}\, f.$$

如果 $\nabla^2 f(x) \succ 0, \forall x \in \mathbf{dom}\, f$，则 f 是严格凸函数.

当函数二阶连续可微时，利用二阶条件判断凸性通常更为方便. 下面给出两个用二阶条件判断凸性的例子.

例 2.5

(1) 考虑二次函数 $f(x) = \frac{1}{2}x^{\mathrm{T}}Px + q^{\mathrm{T}}x + r$ $(P \in \mathcal{S}^n)$，容易计算出其梯度与海瑟矩阵分别为

$$\nabla f(x) = Px + q, \quad \nabla^2 f(x) = P.$$

那么，f 是凸函数当且仅当 $P \succeq 0$.

(2) 考虑最小二乘函数 $f(x) = \frac{1}{2}\|Ax - b\|_2^2$，其梯度与海瑟矩阵分别为

$$\nabla f(x) = A^{\mathrm{T}}(Ax - b), \quad \nabla^2 f(x) = A^{\mathrm{T}}A.$$

注意到 $A^{\mathrm{T}}A$ 恒为半正定矩阵，因此，对任意的 A，f 都是凸函数.

除了上述结果之外，还可以使用上方图 $\mathbf{epi}\, f$ 来判断 f 的凸性. 实际上我们有如下定理：

定理 2.10 *函数 $f(x)$ 为凸函数当且仅当其上方图 $\mathbf{epi}\, f$ 是凸集.*

定理 2.10 的证明留给读者完成.

2.5.3 保凸的运算

要验证一个函数 f 是凸函数，前面已经介绍了三种方法：一是用定义去验证凸性，通常将函数限制在一条直线上；二是利用一阶条件、二阶条件证明函数的凸性；三是直接研究 f 的上方图 $\mathbf{epi}\, f$. 而接下来要介绍的方法说明 f 可由简单的凸函数通过一些保凸的运算得到. 下面的定理说明非负加权和、与仿射函数的复合、逐点取最大值等运算，是不改变函数的凸性的.

定理 2.11

(1) 若 f 是凸函数，则 αf 是凸函数，其中 $\alpha \geqslant 0$.

(2) 若 f_1，f_2 是凸函数，则 $f_1 + f_2$ 是凸函数.

(3) 若 f 是凸函数，则 $f(Ax+b)$ 是凸函数.

(4) 若 $f_1, f_2, \cdots, f_m$ 是凸函数，则 $f(x) = \max\{f_1(x), f_2(x), \cdots, f_m(x)\}$ 是凸函数.

(5) 若对每个 $y \in \mathcal{A}$，$f(x,y)$ 关于 x 是凸函数，则

$$g(x) = \sup_{y\in\mathcal{A}} f(x,y)$$

是凸函数.

(6) 给定函数 $g\colon \mathbb{R}^n \to \mathbb{R}$ 和 $h\colon \mathbb{R} \to \mathbb{R}$，令 $f(x) = h(g(x))$. 若 g 是凸函数，h 是凸函数且单调不减，那么 f 是凸函数；若 g 是凹函数，h 是凸函数且单调不增，那么 f 是凸函数.

(7) 若 $f(x,y)$ 关于 (x,y) 整体是凸函数，C 是凸集，则

$$g(x) = \inf_{y\in C} f(x,y)$$

是凸函数.

证明. 我们只对其中的 (4)(5)(8) 进行证明，剩下的读者可自行验证.

(4) 我们只对 $m = 2$ 的情况验证，一般情况下同理可证. 设

$$f(x) = \max\{f_1(x), f_2(x)\},$$

对任意的 $0 \leqslant \theta \leqslant 1$ 和 $x,y \in \mathbf{dom}\, f$，我们有

$$\begin{aligned} f(\theta x + (1-\theta)y) &= \max\{f_1(\theta x + (1-\theta)y), f_2(\theta x + (1-\theta)y)\} \\ &\leqslant \max\{\theta f_1(x) + (1-\theta)f_1(y), \theta f_2(x) + (1-\theta)f_2(y)\} \\ &\leqslant \theta f(x) + (1-\theta)f(y), \end{aligned}$$

其中第一个不等式是 f_1 和 f_2 的凸性，第二个不等式是将 $f_1(x)$ 和 $f_2(x)$ 放大为 $f(x)$. 所以 f 是凸函数.

(5) 可以直接仿照 (4) 的证明进行验证. 也可利用上方图的性质. 不难看出

$$\mathbf{epi}\, g = \bigcap_{y \in \mathcal{A}} \mathbf{epi}\, f(\cdot, y).$$

由于任意多个凸集的交集还是凸集，所以 $\mathbf{epi}\, g$ 是凸集，根据上方图的性质容易推出 g 是凸函数.

(7) 仍然根据定义进行验证. 任取 $\theta \in (0,1)$ 以及 x_1, x_2，要证

$$g(\theta x_1 + (1-\theta)x_2) \leqslant \theta g(x_1) + (1-\theta)g(x_2).$$

由 g 的定义知对任意 $\varepsilon > 0$，存在 $y_1, y_2 \in C$，使得

$$f(x_i, y_i) < g(x_i) + \varepsilon, \quad i = 1,2.$$

因此

$$\begin{aligned} g(\theta x_1 + (1-\theta)x_2) &= \inf_{y \in C} f(\theta x_1 + (1-\theta)x_2, y) \\ &\leqslant f(\theta x_1 + (1-\theta)x_2, \theta y_1 + (1-\theta)y_2) \\ &\leqslant \theta f(x_1, y_1) + (1-\theta)f(x_2, y_2) \\ &\leqslant \theta g(x_1) + (1-\theta)g(x_2) + \varepsilon, \end{aligned}$$

其中第一个不等号是利用了 C 的凸性，第二个不等号利用了 $f(x,y)$ 的凸性. 最后令 ε 趋于 0 可以得到最终结论. □

2.5.4 凸函数的性质

1. 凸下水平集

凸函数的所有下水平集都为凸集，即有如下结果：

命题 2.4 设 $f(x)$ 是凸函数，则 $f(x)$ 所有的 α-下水平集 C_α 为凸集.

证明. 任取 $x_1, x_2 \in C_\alpha$，对任意的 $\theta \in (0,1)$，根据 $f(x)$ 的凸性我们有

$$\begin{aligned} f(\theta x_1 + (1-\theta)x_2) &\leqslant \theta f(x_1) + (1-\theta) f(x_2) \\ &\leqslant \theta\alpha + (1-\theta)\alpha = \alpha. \end{aligned}$$

这说明 C_α 是凸集. □

需要注意的是，上述命题的逆命题不成立，即任意下水平集为凸集的函数不一定是凸函数. 读者可自行举出反例.

2. 二次下界

强凸函数具有二**次下界**的性质.

引理 2.2 (二次下界) 设 $f(x)$ 是参数为 m 的可微强凸函数，则如下不等式成立：

$$f(y) \geqslant f(x) + \nabla f(x)^{\mathrm{T}}(y-x) + \frac{m}{2}\|y-x\|^2, \quad \forall\, x, y \in \mathbf{dom}\, f. \tag{2.5.1}$$

证明. 由强凸函数的定义，$g(x) = f(x) - \dfrac{m}{2}\|x\|^2$ 是凸函数，根据凸函数的一阶条件可知

$$g(y) \geqslant g(x) + \nabla g(x)^{\mathrm{T}}(y-x),$$

即

$$\begin{aligned} f(y) &\geqslant f(x) - \frac{m}{2}\|x\|^2 + \frac{m}{2}\|y\|^2 + (\nabla f(x) - mx)^{\mathrm{T}}(y-x) \\ &= f(x) + \nabla f(x)^{\mathrm{T}}(y-x) + \frac{m}{2}\|y-x\|^2. \end{aligned}$$

□

利用二次下界容易推出可微强凸函数的下水平集都是有界的，证明留给读者完成.

推论 2.2 设 f 为可微强凸函数，则 f 的所有 α-下水平集有界.

2.6 共轭函数

2.6.1 共轭函数的定义和例子

共轭函数是凸分析中的一个重要概念，其在凸优化问题的理论与算法中扮演着重要角色.

定义 2.19 (共轭函数) *任一适当函数 f 的共轭函数定义为*

$$f^*(y) = \sup_{x \in \mathbf{dom}\, f} \{y^{\mathrm{T}} x - f(x)\}. \tag{2.6.1}$$

设 f 为 $\mathbb{R}$ 上的适当函数，图2.12展示了对固定的 y，$f^*(y)$ 的几何意义. 在这里注意共轭函数是广义实值函数，$f^*(y)$ 可以是正无穷. 自然地，我们

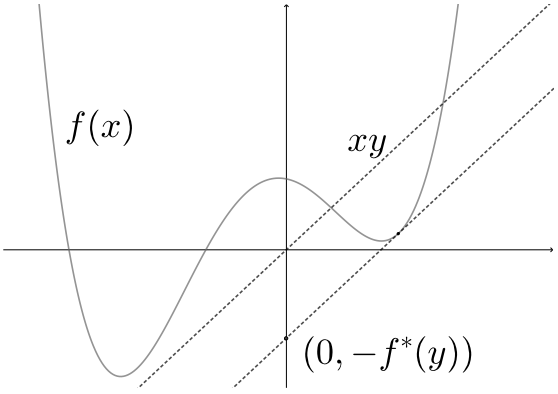


图 2.12 共轭函数

规定其定义域 $\mathbf{dom}\, f^*$ 为使得 $f^*(y)$ 有限的 y 组成的集合. 对任意函数 f 都可以定义共轭函数（不要求 f 是凸的），根据定理 2.11 的 (5)，共轭函数 f^* 恒为凸函数. 由共轭函数的定义，有如下的重要不等式：

命题 2.5 (Fenchel 不等式)

$$f(x) + f^*(y) \geqslant x^{\mathrm{T}} y. \tag{2.6.2}$$

证明. 由定义立即得出，对任意的 $x \in \mathbf{dom}\, f$，

$$f^*(y) = \sup_{x \in \mathbf{dom}\, f} \{y^{\mathrm{T}} x - f(x)\} \geqslant y^{\mathrm{T}} x - f(x),$$

整理即得 (2.6.2) 式. □

以下我们给出一些常见函数的共轭函数.

例 2.6 (二次函数) *考虑二次函数*

$$f(x) = \frac{1}{2} x^{\mathrm{T}} A x + b^{\mathrm{T}} x + c.$$

(1) 强凸情形（$A \succ 0$）：

$$f^*(y) = \frac{1}{2}(y-b)^{\mathrm{T}} A^{-1}(y-b) - c;$$

(2) 一般凸情形（$A \succeq 0$）：

$$f^*(y) = \frac{1}{2}(y-b)^{\mathrm{T}} A^{\dagger}(y-b) - c, \quad \mathbf{dom}\, f^* = \mathcal{R}(A) + b$$

这里 $\mathcal{R}(A)$ 为 A 的像空间.

例 2.7 (凸集的示性函数) 给定凸集 C，其示性函数为

$$I_C(x) = \begin{cases} 0, & x \in C, \\ +\infty, & x \notin C. \end{cases}$$

可知对应的共轭函数为

$$I_C^*(y) = \sup_x \{y^{\mathrm{T}} x - I_C(x)\} = \sup_{x \in C} y^{\mathrm{T}} x.$$

这里 $I_C^*(y)$ 又称为凸集 C 的**支撑函数**.

例 2.8 (范数) 范数的共轭函数为其单位对偶范数球的示性函数，即若 $f(x) = \|x\|$，则

$$f^*(y) = \begin{cases} 0, & \|y\|_* \leqslant 1, \\ +\infty, & \|y\|_* > 1. \end{cases}$$

证明. 对偶范数定义为：$\|y\|_* = \sup\limits_{\|x\| \leqslant 1} x^{\mathrm{T}} y$. 为了计算

$$f^*(y) = \sup_{x \in \mathbf{dom} f} \{y^{\mathrm{T}} x - \|x\|\},$$

我们分两种情形讨论：

(1) 若 $\|y\|_* \leqslant 1$，则 $y^{\mathrm{T}} x \leqslant \|x\|$ 对任一 x 成立，且当 $x = 0$ 时等号成立，从而 $\sup\limits_{x \in \mathbf{dom} f} \{y^{\mathrm{T}} x - \|x\|\} = 0$;

(2) 若 $\|y\|_* > 1$，则至少存在一个 x，使得 $\|x\| \leqslant 1$ 且 $x^{\mathrm{T}} y > 1$，从而对 $t > 0$,

$$f^*(y) \geqslant y^{\mathrm{T}}(tx) - \|tx\| = t(y^{\mathrm{T}} x - \|x\|),$$

而不等式右端当 $t \to +\infty$ 时趋于无穷. □

2.6.2 二次共轭函数

定义 2.20 (二次共轭函数) 任一函数 f 的二次共轭函数定义为

$$f^{**}(x) = \sup_{y \in \mathbf{dom}\, f^*} \{x^{\mathrm{T}} y - f^*(y)\}.$$

显然 f^{**} 恒为闭凸函数，且由 Fenchel 不等式(2.6.2)可知

$$f^{**}(x) \leqslant f(x), \quad \forall x,$$

或等价地，$\mathbf{epi}\, f \subseteq \mathbf{epi}\, f^{**}$. 对于凸函数 f，下面的定理描述了 f 的二次共轭函数与其自身的关系 [112]推论 12.2.1.

定理 2.12 若 f 为闭凸函数，则

$$f^{**}(x) = f(x), \quad \forall x,$$

或等价地，$\mathbf{epi}\, f = \mathbf{epi}\, f^{**}$.

2.7 次梯度

2.7.1 次梯度的定义

前面介绍了可微函数的梯度. 但是对于一般的函数，之前定义的梯度不一定存在. 对于凸函数，类比梯度的一阶性质，我们可以引入次梯度的概念，其在凸优化算法设计与理论分析中扮演着重要角色.

定义 2.21 (次梯度) 设 f 为适当凸函数，x 为定义域 $\mathbf{dom}\, f$ 中的一点. 若向量 $g \in \mathbb{R}^n$ 满足

$$f(y) \geqslant f(x) + g^{\mathrm{T}}(y - x), \quad \forall y \in \mathbf{dom}\, f, \tag{2.7.1}$$

则称 g 为函数 f 在点 x 处的一个**次梯度**. 进一步地，称集合

$$\partial f(x) = \{g \,|\, g \in \mathbb{R}^n, f(y) \geqslant f(x) + g^{\mathrm{T}}(y - x), \forall y \in \mathbf{dom}\, f\} \tag{2.7.2}$$

为 f 在点 x 处的**次微分**.

如图2.13所示，对适当凸函数 $f(x)$，g_1 为点 x_1 处的唯一次梯度，而 g_2, g_3 为点 x_2 处的两个不同的次梯度.

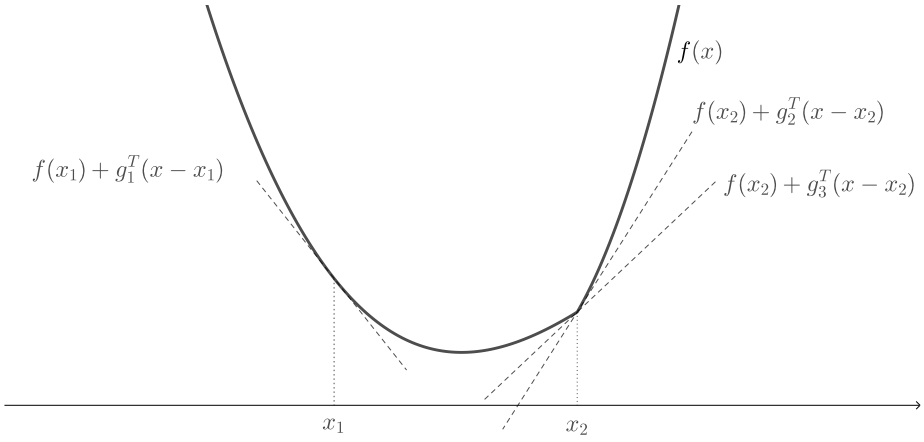


图 2.13 函数 $f(x)$ 的次梯度.

从定义 2.21 可以看出，次梯度实际上借鉴了凸函数判定定理的一阶条件（定理 2.7）. 定义次梯度的初衷之一也是希望它具有类似于梯度的一些性质.

从次梯度的定义可直接推出，若 g 是 $f(x)$ 在 x_0 处的次梯度，则函数

$$l(x) \xlongequal{\text{def}} f(x_0) + g^{\mathrm{T}}(x - x_0)$$

为凸函数 $f(x)$ 的一个全局下界. 此外，次梯度 g 可以诱导出上方图 $\mathbf{epi}\, f$ 在点 $(x, f(x))$ 处的一个支撑超平面，因为容易验证，对 $\mathbf{epi}\, f$ 中的任意点 (y,t)，有

$$\begin{bmatrix} g \\ -1 \end{bmatrix}^{\mathrm{T}} \left(\begin{bmatrix} y \\ t \end{bmatrix} - \begin{bmatrix} x \\ f(x) \end{bmatrix} \right) \leqslant 0, \quad \forall\, (y,t) \in \mathbf{epi}\, f.$$

根据定义可以计算一些简单函数的次微分，在这里我们给出一个例子.

例 2.9 (ℓ_2 范数的次微分) 设 $f(x) = \|x\|_2$，则 $f(x)$ 在点 $x = 0$ 处不可微，我们求其在该点处的次梯度. 注意到对任意的 g 且 $\|g\|_2 \leqslant 1$，根据柯西不等式，

$$g^{\mathrm{T}}(x - 0) \leqslant \|g\|_2 \|x\|_2 \leqslant \|x\|_2 - 0,$$

因此

$$\{g \mid \|g\|_2 \leqslant 1\} \subseteq \partial f(0).$$

接下来说明若 $\|g\|_2 > 1$，则 $g \notin \partial f(0)$. 取 $x = g$，若 g 为次梯度，则

$$\|g\|_2 - 0 \geqslant g^{\mathrm{T}}(g - 0) = \|g\|_2^2 > \|g\|_2,$$

这显然是矛盾的. 综上，我们有

$$\partial f(0) = \{g \mid \|g\|_2 \leqslant 1\}.$$

2.7.2 次梯度的性质

凸函数 $f(x)$ 的次梯度和次微分有许多有用的性质. 下面的定理说明次微分 $\partial f(x)$ 在一定条件下分别为闭凸集和非空有界集.

定理 2.13 设 f 是凸函数，则 $\partial f(x)$ 有如下性质：

(1) 对任何 $x \in \mathbf{dom}\, f$，$\partial f(x)$ 是一个闭凸集（可能为空集）；

(2) 如果 $x \in \mathbf{int}\,\mathbf{dom}\, f$，则 $\partial f(x)$ 非空有界集.

当凸函数 $f(x)$ 在某点处可微时，$\nabla f(x)$ 就是 $f(x)$ 在该点处唯一的次梯度.

命题 2.6 设 $f(x)$ 在 $x_0 \in \mathbf{int}\,\mathbf{dom}\, f$ 处可微，则

$$\partial f(x_0) = \{\nabla f(x_0)\}.$$

证明. 根据可微凸函数的一阶条件（定理 2.7）可知梯度 $\nabla f(x_0)$ 为次梯度. 下证 $f(x)$ 在点 x_0 处不可能有其他次梯度. 设 $g \in \partial f(x_0)$，根据次梯度的定义，对任意的非零 $v \in \mathbb{R}^n$ 且 $x_0 + tv \in \mathbf{dom}\, f, t > 0$ 有

$$f(x_0 + tv) \geqslant f(x_0) + tg^{\mathrm{T}}v.$$

若 $g \neq \nabla f(x_0)$，取 $v = g - \nabla f(x_0) \neq 0$，上式变形为

$$\frac{f(x_0 + tv) - f(x_0) - t\nabla f(x_0)^{\mathrm{T}}v}{t\|v\|} \geqslant \frac{(g - \nabla f(x_0))^{\mathrm{T}}v}{\|v\|} = \|v\|.$$

不等式两边令 $t \to 0$，根据 Fréchet 可微的定义，左边趋于 0，而右边是非零正数，可得到矛盾. □

和梯度类似，凸函数的次梯度也具有某种单调性. 这一性质在很多和次梯度有关的算法的收敛性分析中起到了关键的作用.

定理 2.14 (次梯度的单调性) 设 $f:\mathbb{R}^n\to\mathbb{R}$ 为凸函数，$x,y\in\mathbf{dom}\,f$，则

$$(u-v)^{\mathrm{T}}(x-y)\geqslant 0,$$

其中 $u\in\partial f(x)$，$v\in\partial f(y)$.

证明. 由次梯度的定义，

$$f(y)\geqslant f(x)+u^{\mathrm{T}}(y-x),$$
$$f(x)\geqslant f(y)+v^{\mathrm{T}}(x-y).$$

将以上两个不等式相加即得结论. □

2.7.3 次梯度的计算规则

如何计算一个不可微凸函数的次梯度在优化算法设计中是很重要的问题. 根据定义来计算次梯度一般来说比较繁琐，我们来介绍一些次梯度的计算规则. 本小节讨论的计算规则都默认 $x\in\mathbf{int}\,\mathbf{dom}\,f$.

1. 基本规则

我们首先不加证明地给出一些计算次梯度（次微分）的基本规则.

(1) **可微凸函数：**设 f 为凸函数，若 f 在点 x 处可微，则 $\partial f(x)=\{\nabla f(x)\}$.

(2) **凸函数的非负线性组合：**设 f_1,f_2 为凸函数，且满足

$$\mathbf{int}\,\mathbf{dom}\,f_1\cap\mathbf{dom}\,f_2\neq\varnothing,$$

而 $x\in\mathbf{dom}\,f_1\cap\mathbf{dom}\,f_2$. 若

$$f(x)=\alpha_1 f_1(x)+\alpha_2 f_2(x),\quad \alpha_1,\alpha_2\geqslant 0,$$

则 $f(x)$ 的次微分

$$\partial f(x)=\alpha_1\partial f_1(x)+\alpha_2\partial f_2(x).$$

(3) **线性变量替换：**设 h 为适当凸函数，并且函数 f 满足

$$f(x)=h(Ax+b),\quad \forall x\in\mathbb{R}^m,$$

其中 $A \in \mathbb{R}^{n\times m}$, $b \in \mathbb{R}^n$. 若存在 $x^\sharp \in \mathbb{R}^m$，使得 $Ax^\sharp + b \in \mathbf{int}\,\mathbf{dom}\, h$，则

$$\partial f(x) = A^{\mathrm{T}} \partial h(Ax+b), \quad \forall\, x \in \mathbf{int}\,\mathbf{dom}\, f.$$

注 2.3 第一个结论就是命题 2.6；第二个结论是定理 2.15 的简单推论；第三个结论见 [112]定理 23.9.

2. 两个函数之和的次梯度

以下的 Moreau-Rockafellar 定理给出两个凸函数之和的次微分的计算方法.

定理 2.15 (Moreau-Rockafellar [112]定理 23.8) 设 $f_1, f_2 : \mathbb{R}^n \to (-\infty, +\infty]$ 是两个凸函数，则对任意的 $x_0 \in \mathbb{R}^n$，

$$\partial f_1(x_0) + \partial f_2(x_0) \subseteq \partial(f_1 + f_2)(x_0). \tag{2.7.3}$$

进一步地，若 $\mathbf{int}\,\mathbf{dom} f_1 \cap \mathbf{dom} f_2 \neq \varnothing$，则对任意的 $x_0 \in \mathbb{R}^n$，

$$\partial(f_1 + f_2)(x_0) = \partial f_1(x_0) + \partial f_2(x_0). \tag{2.7.4}$$

3. 函数族的上确界

容易验证一族凸函数的上确界函数仍是凸函数. 我们有如下重要结果：

定理 2.16 (Dubovitskii-Milyutin [41]) 设 $f_1, f_2, \cdots, f_m : \mathbb{R}^n \to (-\infty, +\infty]$ 均为凸函数，令

$$f(x) = \max\{f_1(x), f_2(x), \cdots, f_m(x)\}, \quad \forall x \in \mathbb{R}^n.$$

对 $x_0 \in \bigcap_{i=1}^{m} \mathbf{int}\,\mathbf{dom}\, f_i$，定义 $I(x_0) = \{i \mid f_i(x_0) = f(x_0)\}$，则

$$\partial f(x_0) = \mathbf{conv} \bigcup_{i\in I(x_0)} \partial f_i(x_0). \tag{2.7.5}$$

有了前面的结论，我们可以非常简单地得到一些基本函数的次梯度.

例 2.10 设 f_1, f_2 为凸可微函数(如图 2.14)，令 $f(x) = \max\{f_1(x), f_2(x)\}$.

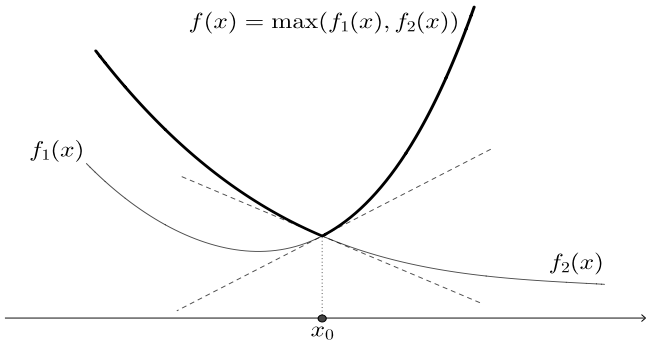


图 2.14 例 2.10 图（一维情形）

(1) 若 $f_1(x) = f_2(x)$，则 $\partial f(x) = \{v \mid v = t\nabla f_1(x) + (1-t)\nabla f_2(x), 0 \leqslant t \leqslant 1\}$；

(2) 若 $f_1(x) > f_2(x)$，则 $\partial f(x) = \{\nabla f_1(x)\}$；

(3) 若 $f_2(x) > f_1(x)$，则 $\partial f(x) = \{\nabla f_2(x)\}$.

例 2.11 (分段线性函数) 令

$$f(x) = \max_{i=1,2,\cdots,m} \{a_i^{\mathrm{T}} x + b_i\},$$

其中 $x, a_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$, $i = 1,2,\cdots,m$，如图 2.15所示，则

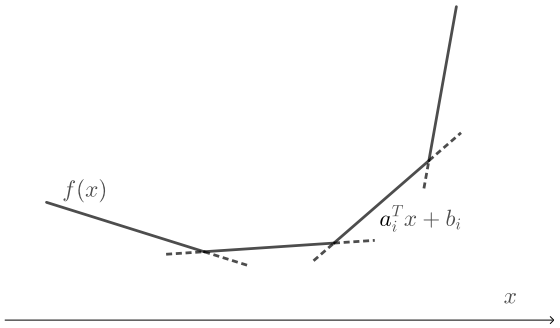
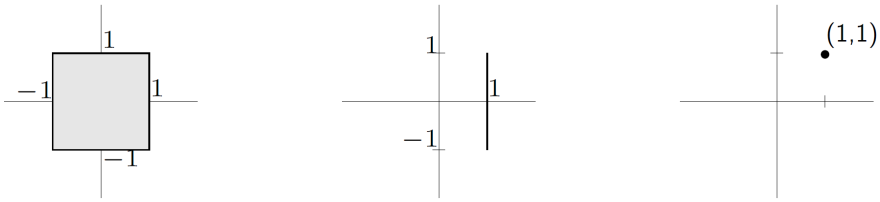


图 2.15 例 2.11 图（一维情形）



$\partial f(0,0) = [-1,1] \times [-1,1]$ $\partial f(1,0) = \{1\} \times [-1,1]$ $\partial f(1,1) = \{(1,1)\}$

图 2.16 ℓ_1 范数的次微分（$n = 2$）

$$\partial f(x) = \mathbf{conv}\{a_i \mid i \in I(x)\},$$

其中

$$I(x) = \{i \mid a_i^{\mathrm{T}} x + b_i = f(x)\}.$$

例 2.12 (ℓ_1 范数) 定义 $f : \mathbb{R}^n \to \mathbb{R}$ 为 ℓ_1 范数，则对 $x = (x_1, x_2, \cdots, x_n) \in \mathbb{R}^n$，有

$$f(x) = \|x\|_1 = \max_{s \in \{-1,1\}^n} s^{\mathrm{T}} x.$$

于是

$$\partial f(x) = J_1 \times J_2 \times \cdots \times J_n, \quad J_k = \begin{cases} [-1,1], & x_k = 0, \\ \{1\}, & x_k > 0, \\ \{-1\}, & x_k < 0. \end{cases}$$

如图 2.16所示.

2.8 总结

本章介绍了本书一些必要的预备知识. 主要内容包括范数、导数、凸分析等方面的内容. 其中凸分析方面的内容编写参考了 [20] 和 Lieven Vandenberghe 教授的课件.

在优化算法实现当中，经常会涉及数值代数方面的内容：例如求解线性方程组、正交分解、特征值（奇异值）分解等运算. 有关数值代数的详细内容，我们推荐读者阅读 [36,143-144].

导数相关内容涉及梯度、海瑟矩阵的基本概念、矩阵变量函数的求导方法. 关于矩阵变量函数的导数的更多内容，可以参考 [103]. 对于 Wirtinger 导数，可以参考 [27] 的第 VI 节.

我们在凸分析部分详细介绍了凸集、凸函数、次梯度的知识，对于共轭函数部分的内容涉及较少，后面章节需要的时候会继续展开讨论. 本章不加证明地给出了凸集、凸函数的很多定理和性质，其中很多证明可以在 [20] 中找到，该书中有对凸集、凸函数进一步的介绍和更多的例子. 比较严格化的凸分析内容，我们建议感兴趣的读者阅读 [112].

习题 2

2.1 证明：矩阵 A 的 2 范数等于其最大奇异值，即

$$\sigma_1(A) = \max_{\|x\|_2=1} \|Ax\|_2.$$

2.2 证明如下有关矩阵范数的不等式：

(a) $\|AB\|_F \leqslant \|A\|_2 \|B\|_F$,

(b) $|\langle A, B\rangle| \leqslant \|A\|_2 \|B\|_*$.

2.3 假设 A 和 B 均为半正定矩阵，求证：$\langle A, B\rangle \geqslant 0$. 提示：利用对称矩阵的特征值分解.

2.4 计算下列矩阵变量函数的导数.

(a) $f(X) = a^{\mathrm{T}} X b$，这里 $X \in \mathbb{R}^{m\times n}$，$a \in \mathbb{R}^m, b \in \mathbb{R}^n$ 为给定的向量；

(b) $f(X) = \mathrm{Tr}(X^{\mathrm{T}} A X)$，其中 $X \in \mathbb{R}^{m\times n}$ 是长方形矩阵，A 是方阵（但不一定对称）；

2.5 考虑二次不等式

$$x^{\mathrm{T}} A x + b^{\mathrm{T}} x + c \leqslant 0,$$

其中 A 为 n 阶对称矩阵，设 C 为上述不等式的解集.

(a) 证明：当 A 正定时，C 为凸集；

(b) 设 C' 是 C 和超平面 $g^{\mathrm{T}} x + h = 0$ 的交集（$g \neq 0$），若存在 $\lambda \in \mathbb{R}$，使得 $A + \lambda g g^{\mathrm{T}}$ 半正定，证明：C' 为凸集.

2.6 利用凸函数二阶条件证明如下结论：

(a) ln-sum-exp 函数：$f(x)=\ln\sum_{k=1}^{n}\exp x_k$ 是凸函数；

(b) 几何平均：$f(x)=(\prod_{k=1}^{n}x_k)^{1/n}$ $(x\in\mathbb{R}^n_{++})$ 是凹函数；

(c) 设 $f(x)=\left(\sum_{i=1}^{n}x_i^p\right)^{1/p}$，其中 $p\in(0,1)$，定义域为 $x>0$，则 $f(x)$ 是凹函数.

2.7 证明定理 2.10.

2.8 求下列函数的共轭函数：

(a) 负熵：$\sum_{i=1}^{n}x_i\ln x_i$；

(b) 矩阵对数：$f(x)=-\ln\det(X)$；

(c) 最大值函数：$f(x)=\max_i x_i$；

(d) 二次锥上的对数函数：$f(x,t)=-\ln(t^2-x^{\mathrm{T}}x)$，注意这里 f 的自变量是 (x,t).

2.9 求下列函数的一个次梯度：

$$f(x)=\|Ax-b\|_2+\|x\|_2.$$

2.10 设 $f(x)$ 为 m-强凸函数，求证：对于任意的 $x\in\mathbf{int}\,\mathbf{dom}\,f$，

$$f(x)-\inf_{y\in\mathbf{dom}\,f}f(y)\leqslant\frac{1}{2m}\mathrm{dist}^2(0,\partial f(x)),$$

其中 $\mathrm{dist}(z,S)$ 表示点 z 到集合 S 的欧几里得距离.

第三章　典型优化问题

在实际中优化问题的形式多种多样. 对于不同种类的优化问题，我们需要根据问题的具体形式，来分析其理论性质以及设计最有效的算法. 本章将会列举一些重要的优化问题. 在这里我们指出，对于同一个实际问题，使用不同的建模手段可能获得形式不同的优化问题. 这些问题的求解难度以及解的性质可能有非常大的差别，因此将实际问题转化为何种优化问题是优化建模中需要重点考虑的.

3.1　线性规划

3.1.1　基本形式和应用背景

线性规划问题的一般形式如下：

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^{\mathrm{T}}x, \\ \text{s.t.} \quad & Ax = b, \\ & Gx \leqslant e, \end{aligned} \tag{3.1.1}$$

其中 $c \in \mathbb{R}^n, A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m, G \in \mathbb{R}^{p\times n}$ 和 $e \in \mathbb{R}^p$ 是给定的矩阵和向量，$x \in \mathbb{R}^n$ 是决策变量. 在实际中，我们考虑问题 (3.1.1) 的两种特殊形式（其他形式都可以转化成这两种形式）：标准形（等式约束和决策变量非负）

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^{\mathrm{T}}x, \\ \text{s.t.} \quad & Ax = b, \\ & x \geqslant 0, \end{aligned} \tag{3.1.2}$$

以及不等式形（没有等式约束）

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^{\mathrm{T}}x, \\ \text{s.t.} \quad & Ax \leqslant b. \end{aligned} \tag{3.1.3}$$

线性规划最先在第二次世界大战时被提出，用于最大化资源的利用效率. 其中的“规划”也是一个军事词汇，指按照既定的时刻表去执行任务或者用最佳方式做人员部署. 线性规划问题的研究很快得到了大家的关注. 二战之后，美国空军启动了一项关于军事规划和分配模型的项目. 在 1947 年，著名的单纯形方法被提出，使得线性规划问题可以被有效地求解. 之后，线性规划用到了更多其他领域当中，如农业、石油、钢铁、运输、通信和运筹学等. 线性规划的有效应用节省了大量的人力、物力和财力. 随着计算机以及求解算法的快速发展，我们可以求解更大规模的线性规划问题，保证了线性规划问题的应用前景.

3.1.2 应用举例

1. 基追踪问题

基追踪问题是压缩感知中的一个基本问题，可以写为

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \|x\|_1, \\ \text{s.t.} \quad & Ax = b. \end{aligned} \tag{3.1.4}$$

对每个 $|x_i|$ 引入一个新的变量 z_i，可以将问题 (3.1.4) 转化为

$$\begin{aligned} \min_{z\in\mathbb{R}^n} \quad & \sum_{i=1}^{n} z_i, \\ \text{s.t.} \quad & Ax = b, \\ & -z_i \leqslant x_i \leqslant z_i, \quad i = 1,2,\cdots,n, \end{aligned}$$

这是一个线性规划问题. 另外，我们也可以引入 x_i 的正部和负部 $x_i^+, x_i^- \geqslant 0$，利用 $x_i = x_i^+ - x_i^-, |x_i| = x_i^+ + x_i^-$，问题 (3.1.4) 的另外一种等价的线性规划形式可以写成

$$\begin{aligned} \min_{x^+,x^-\in\mathbb{R}^n} \quad & \sum_{i=1}^{n} (x_i^+ + x_i^-), \\ \text{s.t.} \quad & Ax^+ - Ax^- = b, \\ & x^+, x^- \geqslant 0. \end{aligned}$$

2. 数据拟合

在数据拟合中，除了常用的最小二乘模型外，还有最小 ℓ_1 范数模型

$$\min_{x\in\mathbb{R}^n} \quad \|Ax-b\|_1, \tag{3.1.5}$$

和最小 ℓ_∞ 范数模型

$$\min_{x\in\mathbb{R}^n} \quad \|Ax-b\|_\infty. \tag{3.1.6}$$

这两个问题都可以转化成线性规划的形式. 对于问题 (3.1.5)，通过引入变量 $y=Ax-b$，可以得到如下等价问题：

$$\begin{aligned} \min_{x,y\in\mathbb{R}^n} \quad & \|y\|_1, \\ \text{s.t.} \quad & y=Ax-b. \end{aligned}$$

利用基追踪问题中类似的技巧，我们可以将上述绝对值优化问题转化成线性规划问题. 对于问题 (3.1.6)，令 $t=\|Ax-b\|_\infty$，则得到等价问题

$$\begin{aligned} \min_{x\in\mathbb{R}^n,\, t\in\mathbb{R}} \quad & t, \\ \text{s.t.} \quad & \|Ax-b\|_\infty \leqslant t. \end{aligned}$$

利用 ℓ_∞ 范数的定义，可以进一步写为

$$\begin{aligned} \min_{x\in\mathbb{R}^n,\, t\in\mathbb{R}} \quad & t, \\ \text{s.t.} \quad & -t\mathbf{1} \leqslant Ax-b \leqslant t\mathbf{1}, \end{aligned}$$

这是一个线性规划问题.

3.2 最小二乘问题

3.2.1 基本形式和应用背景

最小二乘问题的一般形式如下：

$$\min_{x\in\mathbb{R}^n} \quad \sum_{i=1}^{m} r_i^2(x), \tag{3.2.1}$$

其中 $r_i:\mathbb{R}^n\to\mathbb{R}$ 为实值函数. 如果所有的 r_i 都是线性函数，我们称问题 (3.2.1) 为线性最小二乘问题，否则称其为非线性最小二乘问题. 最小二乘问题是线性回归和非线性回归的基础.

最小二乘问题也常用于线性（非线性）方程组问题当中. 当线性（非线性）观测带有噪声时，我们一般会基于该线性（非线性）系统建立最小二乘模型. 特别地，如果噪声服从高斯分布，最小二乘问题的解对应于原问题的最大似然解.

3.2.2 应用举例

1. 线性最小二乘问题

线性最小二乘问题是回归分析中的一个基本模型，它可以表示为

$$\min_{x\in\mathbb{R}^n} \quad \sum_{i=1}^{m}(a_i^{\mathrm{T}}x-b_i)^2,$$

即 $r_i(x)=a_i^{\mathrm{T}}x-b_i$, $i=1,2,\cdots,m$. 记 $A=[a_1,a_2,\cdots,a_m]^{\mathrm{T}}$，那么线性最小二乘问题可以等价地写成如下形式：

$$\min_{x\in\mathbb{R}^n} \quad f(x)\xlongequal{\text{def}}\frac{1}{2}\|Ax-b\|_2^2,$$

这是一个无约束二次目标函数的优化问题. 因为二次函数 f 是凸的，故 $x\in\mathbb{R}^n$ 为其全局极小解当且仅当 x 满足方程

$$\nabla f(x)=A^{\mathrm{T}}(Ax-b)=0.$$

事实上，因为 f 是二次的，我们有

$$\begin{aligned} f(y)&=f(x)+\nabla f(x)^{\mathrm{T}}(y-x)+\frac{1}{2}(y-x)^{\mathrm{T}}\nabla^2 f(x)(y-x)\\ &=f(x)+\nabla f(x)^{\mathrm{T}}(y-x)+\frac{1}{2}(y-x)^{\mathrm{T}}A^{\mathrm{T}}A(y-x). \end{aligned}$$

因此，如果 $\nabla f(x)=0$，根据 $A^{\mathrm{T}}A$ 的半正定性，

$$f(y)\geqslant f(x),\quad \forall y\in\mathbb{R}^n,$$

即 x 为 $f(x)$ 的全局极小解.

反之，如果 $\nabla f(x)\neq 0$，此时我们说明沿着负梯度方向目标函数将减小. 具体地，取 $y=x-t\nabla f(x)$ 且 $t=\dfrac{1}{\lambda_{\max}(A^{\mathrm{T}}A)}$，其中 $\lambda_{\max}(A^{\mathrm{T}}A)$ 表示 $A^{\mathrm{T}}A$

的最大特征值，那么

$$\begin{aligned} f(x - t\nabla f(x)) &\leqslant f(x) - t\|\nabla f(x)\|_2^2 + \frac{1}{2}t^2\lambda_{\max}(A^{\mathrm{T}}A)\|\nabla f(x)\|_2^2 \\ &= f(x) + (-t + \frac{1}{2}t^2\lambda_{\max}(A^{\mathrm{T}}A))\|\nabla f(x)\|_2^2 \\ &= f(x) - \frac{1}{2\lambda_{\max}(A^{\mathrm{T}}A)}\|\nabla f(x)\|_2^2 < f(x). \end{aligned}$$

因而在全局极小解 x 处必有 $\nabla f(x) = 0$.

2. 数据插值

数据插值是数值分析的一个基本问题. 给定数据集 $\{a_i \in \mathbb{R}^p, b_i \in \mathbb{R}^q, i = 1,2,\cdots,m\}$，插值是求一个映射 f，使得

$$b_i = f(a_i), \quad i = 1,2,\cdots,m.$$

在实际中，出于计算上的可行性，我们一般会限制在一个特定函数空间上来求 f 的一个逼近解. 如果利用线性函数逼近，即 $f(a) = Xa + y$，其中 $X \in \mathbb{R}^{q\times p}$, $y \in \mathbb{R}^q$，则为了求解 X, y，可以建立如下最小二乘问题：

$$\min_{X\in\mathbb{R}^{q\times p}} \quad \sum_{i=1}^{m}\|Xa_i + y - b_i\|^2.$$

一般地，假设 $\{\phi_i(a)\}_{i=1}^n (n \leqslant m)$ 为插值空间的一组基，数据插值问题可以写成

$$b_j = f(a_j) = \sum_{i=1}^{n} x_i\phi_i(a_j), \quad j = 1,2,\cdots,m,$$

其中 x_i 为待定系数. 这是关于 x 的线性方程组. 在实际中，我们考虑其替代的最小二乘模型

$$\min_{x\in\mathbb{R}^n} \quad \sum_{j=1}^{m}\|\sum_{i=1}^{n} x_i\phi_i(a_j) - b_j\|^2.$$

对于基 $\phi_i(a)$ 的选取，一般要求其反映数据的物理性质或者内在性质以及计算比较方便.

除了这种基函数的和的方式，深度学习 [81] 也通过一些简单函数的复合来逼近原未知函数. 具体地，假设有一些简单的非线性向量函数 $\phi_i(\theta)$: $\mathbb{R}^q \to \mathbb{R}^q$，并构造如下复合函数：

$$f(\theta) = \phi_n\left(X_n\phi_{n-1}(X_{n-1}\cdots\phi_1(X_1\theta + y_1)\cdots + y_{n-1}) + y_n\right).$$

在实际中常用的简单非线性函数有 ReLU，即

$$\phi_i(\theta) = (\mathrm{ReLU}(\theta_1), \mathrm{ReLU}(\theta_2), \cdots, \mathrm{ReLU}(\theta_q))^{\mathrm{T}}, \quad i = 1,2,\cdots,n,$$

且

$$\mathrm{ReLU}(t) = \begin{cases} t, & t \geqslant 0, \\ 0, & \text{其他}. \end{cases}$$

这样的做法往往会带来更多未知的非线性，因而可能在更大的函数空间中得到未知函数的一个更好的逼近. 将点 a_i 处的取值代入，我们得到如下非线性方程组：

$$\begin{aligned} f(a_i) - b_i = {} & \phi_n\left(X_n\phi_{n-1}(X_{n-1}\cdots\phi_1(X_1a_i + y_1)\cdots + y_{n-1}) + y_n\right) - b_i \\ = {} & 0, \quad i = 1,2,\cdots,n. \end{aligned}$$

这里，需要求解的是关于 $X_1 \in \mathbb{R}^{q\times p}, X_i \in \mathbb{R}^{q\times q}, i = 2,3,\cdots,n$, $y_i \in \mathbb{R}^q, i = 1,2,\cdots,n$ 的非线性方程组. 我们一般考虑替代的最小二乘问题

$$\min_{\{X_i,y_i\}} \quad \sum_{i=1}^{m} \|f(a_i) - b_i\|^2.$$

3.3 复合优化问题

3.3.1 基本形式和应用背景

复合优化问题一般可以表示为如下形式：

$$\min_{x\in\mathbb{R}^n} \quad \psi(x) = f(x) + h(x),$$

其中 $f(x)$ 是光滑函数（比如数据拟合项），$h(x)$ 可能是非光滑的（比如 ℓ_1 范数正则项，约束集合的示性函数，或它们的线性组合）. 从前文介绍的各种各样的应用问题不难发现，复合优化问题在实际中有着重要的应用，并且其中的函数 $h(x)$ 一般都是凸的. 由于应用问题的驱动，复合优化问题的算法近年来得到了大量的研究，比如次梯度法，近似点梯度法，Nesterov 加速算法和交替方向乘子法，等等，我们将在第五、七章中详细地介绍这些算法.

3.3.2 应用举例

1. 图像去噪

图像去噪问题是指从一个带噪声的图像中恢复出不带噪声的原图. 记带噪声的图像为 y，噪声为 ε，那么

$$y = x + \varepsilon,$$

其中 x 为要恢复的真实图像. 图像去噪的一个常用的模型是全变差模型. 考虑图像中相邻位置的像素值存在阶跃的地方是稀疏的，定义全变差算子

$$\|x\|_{TV} = \sum_{i,j} |(\nabla x)_{i,j}|,$$

其中 $(\nabla x)_{i,j}$ 取为 $\sqrt{(x_{i+1,j} - x_{ij})^2 + (x_{i,j+1} - x_{ij})^2}$ 或 $|x_{i+1,j} - x_{ij}| + |x_{i,j+1} - x_{ij}|$，对应的全变差称为是各项同性或者各项异性的. 借助于全变差算子，去噪模型可以表示为

$$\min_{x \in \mathbb{R}^{n_1 \times n_2}} \|x - y\|_F^2 + \lambda \|x\|_{TV},$$

其中目标函数中的第一项是数据保真项，即重构出的图片要与已有的采集信息相容. 第二项是正则项，用来保证重构出的图像的阶跃是稀疏的.

另一类常见的去噪模型基于小波变换. 小波框架是空间中基函数的推广. 设 $W \in \mathbb{R}^{m \times n}$ 为小波框架（$n = n_1 n_2$），这里 m 可以比 n 大，但是有 $\mathrm{rank}(W) = n$. 如果 $m > n$，那么 W 带有一些冗余信息. 对于图像 x，Wx 称为其在小波变换 W 下的小波系数. 在小波变换下，信号尖峰和突变信号对应的小波系数较大，而噪声对应的小波系数的绝对值往往很小. 因此，去噪问题的小波分解模型可以写为

$$\min_x \quad \|\lambda \odot (Wx)\|_1 + \frac{1}{2}\|x - y\|_F^2,$$

其中 $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_m)^{\mathrm{T}}$ 是给定的.

2. 盲反卷积

盲反卷积（也称为去模糊）是图像处理中的一个基本问题，其目的是从一个模糊的图像恢复出原来清晰的图像. 导致图像模糊的原因有很多种，比如相机抖动、聚焦不良，相机和拍摄物体所在的非静止环境以及成像设备的内在缺陷，等等. 因此一般来说盲反卷积是不容易做到的.

为了让这个复杂的问题变得简单一些，我们做如下假设：模糊是线性的（不随图像变化）以及空间不变的（即与像素位置无关）．线性且空间不变的模糊可以表示成一个卷积．令 x 为原始的清晰图像，a 为未知的卷积核对应的矩阵，y 为观测到的模糊图像以及 ε 为观测噪声．盲反卷积问题可以表示成

$$y = a * x + \varepsilon,$$

其中 $*$ 为卷积算子．假设噪声为高斯噪声，则转化为求解优化问题

$$\min_{a,x} \quad \|y - a * x\|_2^2.$$

再假设原始图像信号在小波变换下是稀疏的，我们进一步得到如下复合优化问题：

$$\min_{a,x} \quad \|y - a * x\|_2^2 + \|\lambda \odot (Wx)\|_1,$$

其中 W 是小波框架，$\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_m)^{\mathrm{T}}$ 用来控制稀疏度．

3.4 随机优化问题

3.4.1 基本形式和应用背景

随机优化问题可以表示成以下形式：

$$\min_{x \in \mathcal{X}} \quad \mathbb{E}_{\xi}[F(x, \xi)] + h(x),$$

其中 $\mathcal{X} \subseteq \mathbb{R}^n$ 表示决策变量 x 的可行域，ξ 是一个随机变量（分布一般是未知的）．对于每个固定的 ξ，$F(x, \xi)$ 表示样本 ξ 上的损失或者奖励．正则项 $h(x)$ 用来保证解的某种性质．由于变量 ξ 分布的未知性，其数学期望 $\mathbb{E}_{\xi}[F(x, \xi)]$ 一般是不可计算的．为了得到目标函数值的一个比较好的估计，在实际问题中往往利用 ξ 的经验分布来代替其真实分布．具体地，假设有 N 个样本 $\xi_1, \xi_2, \cdots, \xi_N$，令 $f_i(x) = F(x, \xi_i)$，我们得到优化问题

$$\min_{x \in \mathcal{X}} \quad f(x) \overset{\text{def}}{=\!=} \frac{1}{N} \sum_{i=1}^{N} f_i(x) + h(x), \tag{3.4.1}$$

并称其为经验风险极小化问题或者采样平均极小化问题．这个问题通常是难以求解的，一方面是因为样本数 N 比较多（因此函数值、梯度计算代价比较高），另一方面是因为优化问题的可行域所在空间维数 n 比较大．

这个问题在统计学、机器学习、计算机科学中有着重要的应用. 对于该问题的求解，我们需要在传统的方法上引入随机性以减少算法中目标函数值和梯度等的计算代价，感兴趣的读者可以参考 [57] 的第八章.

3.4.2 应用举例

1. 随机主成分分析

在主成分分析中，如果样本点 ξ 服从某个零均值分布 $\mathcal{D}$，那么找方差最大的 d 维子空间的优化问题可以写成

$$\max_{X\in\mathbb{R}^{p\times d}} \quad \mathrm{Tr}(X^{\mathrm{T}}\mathbb{E}_{\xi\sim\mathcal{D}}[\xi\xi^{\mathrm{T}}]X) \quad \text{s.t.} \quad X^{\mathrm{T}}X = I, \tag{3.4.2}$$

其中 $\mathbb{E}_{\xi\sim\mathcal{D}}[\xi\xi^{\mathrm{T}}]$ 为 ξ 的协方差矩阵. 在实际中，分布 $\mathcal{D}$ 是未知的，已知的只是关于分布 $\mathcal{D}$ 的采样. 比如在在线主成分分析中，样本 ξ_t 是随着时间流逝依次获得的. 这些已有的样本可以看作训练集. 随机主成分分析关心的问题是在求得问题(3.4.2)的高逼近解过程中需要的样本数量以及所消耗的时间. 受制于计算机内存的限制，我们还需要考虑在有限内存情况下的逼近解的计算与分析. 读者可以参考 [3,89].

2. 分布式鲁棒优化

深度学习是机器学习的一个分支，通过利用神经网络来对数据进行表征学习. 深度学习的目的是从已有的未知分布的数据中学出一个好的预测器，其对应优化问题

$$\min_{h} \quad \mathbb{E}_z[F(h,z)],$$

其中预测器 h 是优化变量（见第 1.3 节），并对应于神经网络的参数. 因为数据 z 的真实分布的未知性，我们只有有限的样本点 $z_1,z_2,\cdots,z_n$. 在实际中的一种做法是将这些样本点对应的离散经验分布作为 z 的真实分布，对应的目标函数写成相应的有限和的形式，如第 1.3 节中所述. 这种方式往往保证了在已有样本点上的高预测准确率. 但是，当我们拿到一个新的样本点时，该预测器的准确率可能会下降很多，甚至给出不合理的预测结果. 即预测器的泛化能力较差.

为了提高预测器的泛化能力，另外一种常用的方法是考虑分布式鲁棒优化问题

$$\min_{h} \quad \max_{\hat{z}\in\Gamma} \mathbb{E}_{\hat{z}}[F(h,\hat{z})],$$

这里集合 Γ 中的随机变量的分布与真实数据的分布在一定意义下非常接近. 具体地，在选取 Γ 时，我们需要考虑其对应的实际意义、可解性和数值表现. 给定数据的经验分布，一种方式是通过分布之间的熵来定义分布间的距离，从而定义 Γ 为与经验分布的距离小于给定数的分布的集合. 目前常用的另外一种方式是利用分布间的 Wasserstein 距离，这种距离的好处是其可以改变原来经验分布的支撑集.

3.5 半定规划

半定规划（semidefinite programming, SDP）是线性规划在矩阵空间中的一种推广. 它的目标函数和等式约束均为关于矩阵的线性函数，而它与线性规划不同的地方是其自变量取值于半正定矩阵空间. 作为一种特殊的矩阵优化问题（见第 3.6 节），半定规划在某些结构上和线性规划非常相似，很多研究线性规划的方法都可以作为研究半定规划的基础. 由于半定规划地位的特殊性，我们将在本节中单独讨论半定规划的形式和应用. 而一般的矩阵优化问题将在第 3.6 节中讨论.

3.5.1 基本形式和应用背景

半定规划问题的一般形式如下：

$$\begin{aligned} \min \quad & c^{\mathrm{T}}x, \\ \text{s.t.} \quad & x_1A_1 + x_2A_2 + \cdots + x_nA_n + B \preceq 0, \\ & Gx = h, \end{aligned} \tag{3.5.1}$$

其中 $c \in \mathbb{R}^n, A_i \in \mathcal{S}^m, i = 1,2,\cdots,m, B \in \mathcal{S}^m, G \in \mathbb{R}^{p\times n}, h \in \mathbb{R}^p$ 为已知的向量和矩阵，$x = (x_1, x_2, \cdots, x_n) \in \mathbb{R}^n$ 是自变量. 这里，如果矩阵 A_i, B 是对角的，那么问题 (3.5.1) 退化为线性规划问题. 类似于线性规划问题，我们考虑半定规划的标准形式

$$\begin{aligned} \min \quad & \langle C, X\rangle, \\ \text{s.t.} \quad & \langle A_1, X\rangle = b_1, \\ & \cdots \\ & \langle A_m, X\rangle = b_m, \\ & X \succeq 0, \end{aligned} \tag{3.5.2}$$

和不等式形式

$$\begin{aligned} \min \quad & c^{\mathrm{T}}x, \\ \text{s.t.} \quad & x_1A_1 + x_2A_2 + \cdots + x_nA_n + B \preceq 0. \end{aligned} \tag{3.5.3}$$

形如(3.5.1) 式的优化问题都可以转化成 (3.5.2) 式或者 (3.5.3) 式的形式.

1995 年，文章 [54] 开创性地运用半定规划提出了一个求解 NP 难的最大割问题的 0.8786-近似算法. 半定规划被认为是自 20 世纪 50 年代著名的线性规划以后的另一个数学规划领域革命性的研究进展. 半定规划的发展得益于线性规划的充分研究. 尽管特殊的半定规划可看成线性规划，但作为一类特殊的矩阵优化问题，半定规划具有不同于经典线性与非线性优化问题的特点——其约束集合不是多面体. 在实际应用方面，半定规划和线性规划一样，作为重要的凸优化量化建模工具被应用于工程、经济学等领域.

3.5.2 应用举例

1. 二次约束二次规划问题的半定规划松弛

考虑二次约束二次规划问题

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & x^{\mathrm{T}}A_0x + 2b_0^{\mathrm{T}}x + c_0, \\ \text{s.t.} \quad & x^{\mathrm{T}}A_ix + 2b_i^{\mathrm{T}}x + c_i \leqslant 0, \quad i = 1,2,\cdots,m, \end{aligned} \tag{3.5.4}$$

其中 A_i 为 $n \times n$ 对称矩阵. 当部分 A_i 为对称不定矩阵时，问题 (3.5.4) 是 NP 难的非凸优化问题.

现在我们写出问题(3.5.4) 的半定规划松弛问题. 对任意 $x \in \mathbb{R}^n$ 以及 $A \in \mathcal{S}^n$，有恒等式

$$x^{\mathrm{T}}Ax = \mathrm{Tr}(x^{\mathrm{T}}Ax) = \mathrm{Tr}(Axx^{\mathrm{T}}) = \left\langle A, xx^{\mathrm{T}} \right\rangle,$$

因此问题(3.5.4) 中所有的二次项均可用下面的方式进行等价刻画：

$$x^{\mathrm{T}}A_ix + 2b_i^{\mathrm{T}}x + c_i = \left\langle A_i, xx^{\mathrm{T}} \right\rangle + 2b_i^{\mathrm{T}}x + c_i.$$

所以，原始问题等价于

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \langle A_0, X\rangle + 2b_0^{\mathrm{T}}x + c_0 \\ \text{s.t.} \quad & \langle A_i, X\rangle + 2b_i^{\mathrm{T}}x + c_i \leqslant 0, \quad i = 1,2,\cdots,m, \\ & X = xx^{\mathrm{T}}. \end{aligned} \tag{3.5.5}$$

进一步地，

$$x^{\mathrm{T}}A_i x + 2b_i^{\mathrm{T}}x + c_i = \left\langle \begin{pmatrix} A_i & b_i \\ b_i^{\mathrm{T}} & c_i \end{pmatrix}, \begin{pmatrix} X & x \\ x^{\mathrm{T}} & 1 \end{pmatrix} \right\rangle, \\ \xlongequal{\text{def}} \langle \overline{A_i}, \overline{X} \rangle, \quad i = 0,1,\cdots,m.$$

接下来将等价问题 (3.5.5) 松弛为半定规划问题. 在问题 (3.5.5) 中，唯一的非线性部分是约束 $X = xx^{\mathrm{T}}$，我们将其松弛成半正定约束 $X \succeq xx^{\mathrm{T}}$. 可以证明，$\overline{X} \succeq 0$ 与 $X \succeq xx^{\mathrm{T}}$ 是等价的. 因此这个问题的半定规划松弛可以写成

$$\begin{aligned} \min \quad & \langle \overline{A_0}, \overline{X} \rangle \\ \text{s.t.} \quad & \langle \overline{A_i}, \overline{X} \rangle \leqslant 0, \quad i = 1,2,\cdots,m, \\ & \overline{X} \succeq 0, \\ & \overline{X}_{n+1,n+1} = 1. \end{aligned}$$

其中“松弛”来源于我们将 $X = xx^{\mathrm{T}}$ 替换成了 $X \succeq xx^{\mathrm{T}}$.

2. 最大割问题的半定规划松弛

令 G 为一个无向图，其节点集合为 $V = \{1,2,\cdots,n\}$ 和边的集合为 E. 令 $w_{ij} = w_{ji}$ 为边 $(i,j) \in E$ 上的权重，并假设 $w_{ij} \geqslant 0, (i,j) \in E$. 最大割问题是找到节点集合 V 的一个子集 S 使得 S 与它的补集 $\overline{S}$ 之间相连边的权重之和最大化. 我们可以将最大割问题写成如下整数规划的形式：令 $x_j = 1$, $j \in S$ 和 $x_j = -1$, $j \in \overline{S}$，则

$$\begin{aligned} \max \quad & \frac{1}{2}\sum_{i<j}(1 - x_i x_j)w_{ij} \\ \text{s.t.} \quad & x_j \in \{-1,1\},\ j = 1,2,\cdots,n. \end{aligned} \tag{3.5.6}$$

在问题 (3.5.6) 中，只有当 x_i 与 x_j 不同时，目标函数中 w_{ij} 的系数非零. 最大割问题是一个离散优化问题，很难在多项式时间内找到它的最优解. 接下来介绍如何将问题 (3.5.6) 松弛成一个半定规划问题.

令 $W = (w_{ij}) \in \mathcal{S}^n$，并定义 $C = -\frac{1}{4}(\mathrm{Diag}(W\mathbf{1}) - W)$ 为图 G 的拉普拉斯矩阵的 $-\frac{1}{4}$ 倍，则问题 (3.5.6) 可以等价地写为

$$\begin{aligned} \min \quad & x^{\mathrm{T}}Cx, \\ \text{s.t.} \quad & x_i^2 = 1,\ i = 1,2,\cdots,n. \end{aligned}$$

由于目标函数是关于 x 的二次函数，利用前面的技巧可将其等价替换为 $\langle C, xx^{\mathrm{T}} \rangle$. 接下来令 $X = xx^{\mathrm{T}}$，注意到约束 $x_i^2 = 1$，这意味着矩阵 X 对角线元素 $X_{ii} = 1$. 因此利用矩阵形式我们将最大割问题转化为

$$\begin{aligned} \min \quad & \langle C, X \rangle, \\ \text{s.t.} \quad & X_{ii} = 1,\ i = 1, 2, \cdots, n, \\ & X \succeq 0,\ \mathrm{rank}(X) = 1. \end{aligned} \tag{3.5.7}$$

问题 (3.5.7) 和 (3.5.6) 是等价的，这是因为 $X = xx^{\mathrm{T}}$ 可以用约束 $X \succeq 0$ 和 $\mathrm{rank}(X) = 1$ 等价刻画. 若在问题 (3.5.7) 中将秩一约束去掉，我们就可以得到最大割问题的半定规划松弛形式

$$\begin{aligned} \min \quad & \langle C, X \rangle \\ \text{s.t.} \quad & X_{ii} = 1,\ i = 1, 2, \cdots, n, \\ & X \succeq 0. \end{aligned} \tag{3.5.8}$$

问题 (3.5.8) 是原最大割问题的一个松弛，因此它们并不等价. 文献 [54] 指出求解问题 (3.5.8) 可以给出原最大割问题的一个 0.8786-近似解.

3.6 矩阵优化

3.6.1 基本形式和应用背景

矩阵优化问题具有如下形式：

$$\min_{X \in \mathcal{X}} \quad \psi(X),$$

其中 $\mathcal{X}$ 为特定的矩阵空间，$\psi(X) : \mathcal{X} \to \mathbb{R}$ 为给定的函数，可能是非光滑的. 对于矩阵优化问题，如果决策变量为一个 $n \times n$ 矩阵，那么我们可能需要确定 n^2 个元素. 因此，决策变量的维数过大往往是矩阵优化问题难以快速求解的一个重要原因.

矩阵优化是在近几十年发展起来的一类变量含有矩阵的优化问题. 它广泛地出现在组合数学、材料科学、机器学习和统计学等各种各样的应用当中. 和向量相比，矩阵有许多新的性质：例如秩、特征值等. 所以矩阵优化问题的求解通常要困难一些.

3.6.2 应用举例

1. 非负矩阵分解

假设 a 为 d 维空间中的非负随机向量，其 n 个观测值为 $\{a_i\}_{i=1}^n$. 记矩阵 $A=[a_1,a_2,\cdots,a_n]\in\mathbb{R}^{d\times n}$，非负矩阵分解问题是指将 A 分解成非负 $d\times p$ 基矩阵 $X=[x_1,x_2,\cdots,x_p]$ 和非负 $p\times n$ 系数矩阵 $Y=[y_1,y_2,\cdots,y_n]$ 的乘积，即

$$A=XY.$$

从上面的表达式可以看出，y_j 为观测点 a_j 在基矩阵 X 上的权重系数. 也就是说，非负矩阵分解把数据分成基向量的线性组合. 通常选取 $p\ll d$，那么得到的基矩阵 X 的列张成了原数据空间的一个子空间. 这本质上是将高维空间中的数据在一个低维空间中表示. 当数据点的内蕴结构完全被基矩阵 X 包含时，我们就得到了一个很好的低维表示.

一般情况下，由于观测含有噪声，原始数据矩阵 A 和分解 XY 不会完全吻合. 在这种情况下我们应当寻找误差最小的解. 利用矩阵的 F 范数可以定义相似性度量

$$\|A-XY\|_F^2,$$

我们考虑如下优化问题

$$\min_{X\in\mathbb{R}^{d\times p},\,Y\in\mathbb{R}^{p\times n}} \quad \|A-XY\|_F^2,\quad \text{s.t.}\quad X\geqslant 0,\,Y\geqslant 0, \tag{3.6.1}$$

其中“$\geqslant 0$”表示矩阵的每个元素是非负的.

从低维空间逼近的角度来看，非负矩阵分解模型和主成分分析模型类似. 但在实际问题中，非负矩阵分解模型会得到比主成分分析模型更有实际意义的解. 比如，给定很多幅人脸图片（都可以用元素值为 $0\sim 255$ 的矩阵来表示其灰度图），我们想要提取脸部的特征. 利用主成分分析得到的主成分可能包含负数像素值，这是不合理的. 但是如果使用非负矩阵分解，则可以有效避免这类情形的发生.

我们称问题 (3.6.1) 为基本的非负矩阵分解模型. 根据具体应用的不同，有时还考虑带正则项的非负矩阵分解模型

$$\begin{aligned}\min_{X\in\mathbb{R}^{d\times p},\,Y\in\mathbb{R}^{p\times n}} \quad & \|A-XY\|_F^2+\alpha r_1(X)+\beta r_2(Y),\\ \text{s.t.}\quad & X\geqslant 0,\quad Y\geqslant 0,\end{aligned} \tag{3.6.2}$$

其中 $r_1(X)$ 和 $r_2(Y)$ 是正则项，$\alpha,\beta>0$ 是用来权衡拟合项和正则项的正则化参数. 比如，如果 Y 的列是稀疏的，那么每一个观测值都可以用少数几个基向量来表示. 相应地，我们可以惩罚 Y 的每一列的 ℓ_1 范数. 为了保证基向量的线性无关性，往往还要求 X 的列之间是相互正交的. 此外，如果数据矩阵 A 分布在一个低维的非线性流形上，则考虑流形或者图上的非负矩阵分解模型. 关于更多非负矩阵分解模型的介绍，读者可以参考 [125].

3.7 优化模型语言

模型语言的发展开始于 19 世纪 70 年代后期，其主要动因是计算机的出现. 在优化模型语言中，优化模型可以写成和数学表达式很类似的方式，以此给用户带来更便捷的服务. 模型的表达式形式是与求解器无关的，不同的求解器需要用优化模型语言将给定的模型和数据转为其求解的标准形式，然后再对其进行求解. 这类工具有三个优点：一是将容易出错的转化步骤交给计算机完成，降低错误率；二是在模型和算法之间建立了一个清晰的界限；三是对于困难的问题，可以尝试不用的求解器，得到更好的结果.

本节主要介绍 CVX 这一工具，它是一个以 MATLAB 为基础的优化模型语言，用来求解凸优化问题. 它允许将优化问题的目标函数以及约束用 MATLAB 语法来写. 比如考虑如下优化问题：

$$\begin{aligned} \min \quad & \|Ax-b\|_2, \\ \text{s.t.} \quad & Cx=d, \\ & \|x\|_\infty \leqslant e, \end{aligned}$$

它可以写成

```
m = 20; n = 10; p = 4;
A = randn(m,n); b = randn(m,1);
C = randn(p,n); d = randn(p,1); e = rand;
cvx_begin
    variable x(n)
    minimize( norm( A * x - b, 2 ) )
    subject to
    C * x == d
    norm( x, Inf ) <= e
```

```
cvx_end
```

代码中的前三行是关于 A,b,C,d,e 的构造．在调用 CVX 求解的时候，对应的代码需要以 `cvx_begin` 开始，并且以 `cvx_end` 结尾．在这两行语句之间，我们需要定义要求解的优化问题．在上面的例子中，`variable x(n)` 表示决策变量 x 为 n 维空间中的向量．目标函数 $\|Ax-b\|_2$ 则用 `norm( A * x - b, 2 )` 来表示，`minimize` 表示求解目标函数的极小值．最后以 `subject to` 开始描述问题的约束，`C * x == d` 和 `norm( x, Inf ) <= e` 分别表示约束 $Cx=d$ 和 $\|x\|_\infty \leqslant e$．执行上述代码，CVX 会选取默认的凸优化问题算法来返回上面问题的解．

CVX 采用了一种快速构造和识别凸性的准则，服从这个准则的凸问题都可以很快地被识别出来．之后 CVX 根据用户的选择调用已有软件包来求解变形后的凸优化问题，这些软件包括免费软件 SDPT3 和 SeDuMi 以及商业软件 Gurobi 和 MOSEK 等．除了一些典型问题外，CVX 还可以识别一些更复杂的凸优化问题，例如带 ℓ_1 范数的优化问题．更多内容可以参考 [59]．目前 CVX 还有 Julia 语言版本 [124] 和 Python 语言版本 CVXPY [40]．除 CVX 外，还有很多发展成熟的优化模型语言可供我们使用，如 AMPL（a mathematical programming language）[47]，YALMIP [85] 等．感兴趣的读者可自行了解相关内容．

3.8 总结

本章介绍了常见的最优化问题以及具体实例．

线性规划作为动态规划问题的一种转化形式，其对于动态规划问题的理论与算法设计有着重要的参考意义．线性规划在管理、最优运输等领域中有着各式各样的应用，感兴趣的读者可以参考 [121]．

由于压缩感知的巨大成功，复合优化的研究得到了人们大量的关注．由于问题的解的稀疏性或者低秩性，通过有效地添加正则项，可以从理论上保证复合优化问题的解满足我们想要的性质．本章介绍的复合优化问题的正则项都是凸的，但是在实际中往往一些非凸正则项对应的模型能够更好地逼近原问题，相关内容可以参考 [126]．随着大量复合优化问题的提出，复合优化问题的算法也得到了大量的研究，我们会在后续章节中详细介绍它们．

矩阵优化是向量优化的推广，其中一个典型问题形式是半定规划．因为半定规划的良好的理论性质，并且得益于内点法的有效求解，人们倾向于建

立半定规划模型或利用半定规划来松弛已有的非凸优化模型. 但是当半定规划对应的半定矩阵维数很高时，现有的算法也很难求解. 最新的一些研究则是考虑用分解模型来逼近半定规划问题，从而有效地求解. 对于聚类问题中的优化问题，其决策变量可以表达为置换矩阵，因为相应约束的复杂性，人们经常通过松弛来进行逼近求解. 考虑到半定规划松弛后仍难以求解，人们提出了一些更有效的松弛方式，相关内容可以参考 [139].

随机优化的复兴得益于数据时代的到来. 为了使得模型具有更好的泛化能力，建立的随机模型不仅要在已有的数据上表现良好，而且也要在未被采集到的数据上性能优异. 那么关于数据的分布的估计，以及如何利用已有的数据衍生出更多的数据来增加模型的鲁棒性是人们在建立随机模型的时候考虑的重点，一个比较著名的工作可参考 GAN [58].

在实际问题中还有一类重要的优化问题的变量落在某种微分流形上，称为流形优化，读者可以参考 [1]. 对于优化模型语言，常用的还有 YALMIP [85].对于半定规划问题,最新的一些软件包有 SDPNAL[141],SDPNAL+[133] 和 SSNSDP [80]. 对于流形优化，软件包有 OptM [127]，Manopt [18] 和 ARNT [73].

习题 4

3.1 将下面的问题转化为线性规划：给定 $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$,

(a) $\min\limits_{x\in\mathbb{R}^n} \quad \|Ax-b\|_1, \quad \text{s.t.} \quad \|x\|_\infty \leqslant 1;$

(b) $\min\limits_{x\in\mathbb{R}^n} \quad \|x\|_1, \quad \text{s.t.} \quad \|Ax-b\|_\infty \leqslant 1;$

(c) $\min\limits_{x\in\mathbb{R}^n} \quad \|Ax-b\|_1 + \|x\|_\infty;$

(d) $\min\limits_{x\in\mathbb{R}^n} \quad \sum\limits_{i=1}^{m} \max\{0, a_i^{\mathrm{T}}x + b_i\}.$

3.2 求解下面的线性规划问题：给定向量 $c \in \mathbb{R}^n$,

(a) $\min\limits_{x\in\mathbb{R}^n} \quad c^{\mathrm{T}}x, \quad \text{s.t.} \quad 0 \leqslant x \leqslant \mathbf{1};$

(b) $\min\limits_{x\in\mathbb{R}^n} \quad c^{\mathrm{T}}x, \quad \text{s.t.} \quad -\mathbf{1} \leqslant x \leqslant \mathbf{1};$

(c) $\min\limits_{x\in\mathbb{R}^n} \quad c^{\mathrm{T}}x, \quad \text{s.t.} \quad -1 \leqslant \mathbf{1}^{\mathrm{T}}x \leqslant 1;$

(d) $\min\limits_{x\in\mathbb{R}^n} \quad c^{\mathrm{T}}x, \quad \text{s.t.} \quad \mathbf{1}^{\mathrm{T}}x = 1, \quad x \geqslant 0;$

3.3 在数据插值中，考虑一个简单的复合模型（取 ϕ 为恒等映射，两层复合模型）：

$$\min_{X_1\in\mathbb{R}^{q\times p},X_2\in\mathbb{R}^{q\times q}} \quad \sum_{i=1}^{m}\|X_2X_1a_i-b_i\|_2^2,$$

其中 $a_i\in\mathbb{R}^p,b_i\in\mathbb{R}^q,i=1,2,\cdots,m.$

(a) 试计算目标函数关于 X_1,X_2 的导数；

(b) 试给出该问题的最优解.

3.4 给定数据点 $a_i\in\mathbb{R}^n,b_i\in\mathbb{R}^n,i=1,2,\cdots,m$，我们用二次函数拟合，即求 $X\in\mathcal{S}^n,y\in\mathbb{R}^n,z\in\mathbb{R}$ 使得

$$b_i\approx f(a_i)=a_i^{\mathrm{T}}Xa_i+y^{\mathrm{T}}a_i+z,\quad i=1,2,\cdots,m.$$

这里假设数据点 $a_i\in\mathcal{B}=\{a\in\mathbb{R}^n\mid l\leqslant a\leqslant u\}$. 相应的最小二乘模型为

$$\sum_{i=1}^{m}(f(a_i)-b_i)^2.$$

此外，对函数 f 有三个额外要求：(1) f 是凹函数；(2) f 在集合 $\mathcal{B}$ 上是非负的，即 $f(a)\geqslant 0,\forall a\in\mathcal{B}$；(3) f 在 $\mathcal{B}$ 上是单调非减的，即对任意的 $a,\hat{a}\in\mathcal{B}$ 且满足 $a\leqslant\hat{a}$，有 $f(a)\leqslant f(\hat{a})$.

请将上述问题表示成一个凸优化问题，并尽可能地简化.

3.5 考虑下面的复合优化问题：

$$\min_{x\in\mathbb{R}^n} \quad \|x\|_1+\|Dx-a\|_2^2,$$

其中 $a\in\mathbb{R}^n,D=\mathrm{Diag}(d_1,d_2,\cdots,d_n)$ 均已知. 试给出最优解 x^* 的表达式.

3.6 考虑下面的复合优化问题：

$$\min_{x\in\mathbb{R}^n} \quad \|x\|_0+\|Dx-a\|_2^2,$$

其中 $a\in\mathbb{R}^n,D=\mathrm{Diag}(d_1,d_2,\cdots,d_n)$ 均已知. 试给出最优解 x^* 的表达式.

3.7 将不等式形式的半定规划问题 (3.5.1) 转化成标准形式 (3.5.2).

3.8 对于对称矩阵 $C \in \mathcal{S}^n$，记其特征值分解为 $C = \sum_{i=1}^{n} \lambda_i u_i u_i^{\mathrm{T}}$，假设

$$\lambda_1 \geqslant \cdots \geqslant \lambda_m > 0 > \lambda_{m+1} \geqslant \cdots \geqslant \lambda_n,$$

考虑如下半定规划问题：

$$\begin{aligned} \min_{X \in \mathcal{S}^n} \quad & \langle C, X \rangle, \\ \text{s.t.} \quad & u_i^{\mathrm{T}} X u_i = 0, i = m+1, m+2, \cdots, n, \\ & X \succeq 0. \end{aligned}$$

试给出该问题最优解的表达式.

3.9 如果在最大割问题(3.5.6)中，约束 $x_j \in \{-1, 1\}$ 改为 $x_j \in \{0, 1\}$，即对应优化问题

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{i<j} w_{ij}(1 - x_i x_j), \\ \text{s.t.} \quad & x_j \in \{0, 1\},\ j = 1, 2, \cdots, n. \end{aligned}$$

试给出其一个半定规划松弛.

第四章　最优性理论

正如第三章所介绍的，在实际中最优化问题的形式多种多样．给定一类具体的优化问题，我们首先需要分析其解的存在性．如果优化问题的解存在，再考虑如何设计算法求出其最优解．一般的非凸优化问题可能存在很多局部极小解，但其往往也能够满足实际问题的要求．对于这些局部（全局）极小解的求解，最优性理论是至关重要的．

4.1　最优化问题解的存在性

考虑优化问题 (1.1.1)

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x), \\ \text{s.t.} \quad & x \in \mathcal{X}, \end{aligned} \tag{4.1.1}$$

其中 $\mathcal{X} \subseteq \mathbb{R}^n$ 为可行域．对于问题 (4.1.1)，首先要考虑的是最优解的存在性，然后考虑如何求出其最优解．在数学分析课程中，我们学习过 Weierstrass 定理，即定义在紧集上的连续函数一定存在最大 (最小) 值点．而在许多实际问题中，定义域可能不是紧的，目标函数也不一定连续，因此需要将此定理推广来保证最优化问题解的存在性．

定理 4.1 (Weierstrass 定理) *考虑一个适当且闭的函数 $f: \mathcal{X} \to (-\infty, +\infty]$，假设下面三个条件中任意一个成立：*

(1) $\mathbf{dom} f \xlongequal{\text{def}} \{x \in \mathcal{X} : f(x) < +\infty\}$ *是有界的；*

(2) 存在一个常数 $\bar{\gamma}$ 使得下水平集

$$C_{\bar{\gamma}} \xlongequal{\text{def}} \{x \in \mathcal{X} : f(x) \leqslant \bar{\gamma}\}$$

是非空且有界的；

(3) f 是强制的，即对于任一满足 $\|x^k\| \to +\infty$ 的点列 $\{x^k\} \subset \mathcal{X}$，都有

$$\lim_{k\to\infty} f(x^k) = +\infty,$$

那么，问题 (4.1.1) 的最小值点集 $\{x \in \mathcal{X} \mid f(x) \leqslant f(y), \forall y \in \mathcal{X}\}$ 是非空且紧的.

证明. 假设条件 (2) 成立. 我们先证下确界 $t \overset{\text{def}}{=\!=} \inf_{x\in\mathcal{X}} f(x) > -\infty$. 采用反证法. 假设 $t = -\infty$，则存在点列 $\{x^k\}_{k=1}^{\infty} \subset C_{\bar\gamma}$，使得 $\lim_{k\to\infty} f(x^k) = t = -\infty$. 因为 $C_{\bar\gamma}$ 的有界性，点列 $\{x^k\}$ 一定存在聚点，记为 x^*. 根据上方图的闭性，我们知道 $(x^*, t) \in \mathbf{epi}\, f$，即有 $f(x^*) \leqslant t = -\infty$. 这与函数的适当性矛盾，故 $t > -\infty$.

利用上面的论述，我们知道 $f(x^*) \leqslant t$. 因为 t 是下确界，故必有 $f(x^*) = t$. 这就证明了下确界是可取得的. 再根据定理 2.2以及 $C_{\bar\gamma}$ 的有界性，易知最小值点集是紧的.

假设条件 (1) 成立，则 $\mathbf{dom}\, f$ 是有界的. 因为 f 是适当的，即存在 $x_0 \in \mathcal{X}$ 使得 $f(x_0) < +\infty$. 令 $\bar\gamma = f(x_0)$，则下水平集 $C_{\bar\gamma}$ 是非空有界的，那么利用条件 (2) 的结论，可知问题 (4.1.1) 的最小值点集是非空且紧的.

假设条件 (3) 成立. 我们沿用上面定义的 x_0, $\bar\gamma \overset{\text{def}}{=\!=} f(x_0)$ 以及下水平集 $C_{\bar\gamma}$. 因为 f 是强制的，则 $C_{\bar\gamma}$ 是非空有界的（假设无界，则存在点列 $\{x^k\} \subset C_{\bar\gamma}$ 满足 $\lim_{k\to\infty} \|x^k\| = +\infty$，由强制性有 $\lim_{k\to\infty} f(x^k) = +\infty$，这与 $f(x) \leqslant \bar\gamma$ 矛盾），即推出条件 (2). □

定理 4.1 的三个条件在本质上都是保证 $f(x)$ 的最小值不能在无穷远处取到，因此我们可以仅在一个有界的下水平集中考虑 $f(x)$ 的最小值. 同时要求 $f(x)$ 为适当且闭的函数，并不需要 $f(x)$ 的连续性. 因此定理4.1比数学分析中的 Weierstrass 定理应用范围更广.

当定义域不是有界闭集时，我们通过例子来进一步解释上面的定理. 对于强制函数 $f(x) = x^2$, $x \in \mathcal{X} = \mathbb{R}$，其全局最优解一定存在. 但对于适当且闭的函数 $f(x) = e^{-x}$, $x \in \mathcal{X} = \mathbb{R}$，它不满足定理4.1三个条件中任意一个，因此我们不能断言其全局极小值点存在. 事实上，其全局极小值点不存在.

4.2 无约束可微问题的最优性理论

无约束可微优化问题通常表示为如下形式：

$$\min_{x\in\mathbb{R}^n} \quad f(x), \tag{4.2.1}$$

其中假设 f 是连续可微函数. 第一章已经引入了局部最优解和全局最优解的定义. 给定一个点 $\bar{x}$，我们想要知道这个点是否是函数 f 的一个局部极小解或者全局极小解. 如果从定义出发，需要对其邻域内的所有点进行判断，这是不可行的. 因此，需要一个更简单的方式来验证一个点是否为极小值点. 我们称其为最优性条件，它主要包含一阶最优性条件和二阶最优性条件.

4.2.1 一阶最优性条件

一阶最优性条件是利用梯度（一阶）信息来判断给定点的最优性. 这里先考虑目标函数可微的情形，并给出下降方向的定义.

定义 4.1 (下降方向) *对于可微函数 f 和点 $x\in\mathbb{R}^n$，如果存在向量 d 满足*

$$\nabla f(x)^{\mathrm{T}} d < 0,$$

那么称 d 为 f 在点 x 处的一个 **下降方向**.

由下降方向的定义，容易验证: 如果 f 在点 x 处存在一个下降方向 d，那么对于任意的 $T>0$，存在 $t\in(0,T]$，使得

$$f(x+td) < f(x).$$

因此，在局部最优点处不能有下降方向. 我们有如下一阶必要条件：

定理 4.2 (一阶必要条件) *假设 f 在全空间 $\mathbb{R}^n$ 可微. 如果 x^* 是一个局部极小点，那么*

$$\nabla f(x^*) = 0.$$

证明. 任取 $v\in\mathbb{R}^n$，考虑 f 在点 $x=x^*$ 处的泰勒展开

$$f(x^*+tv) = f(x^*) + tv^{\mathrm{T}}\nabla f(x^*) + o(t),$$

整理得

$$\frac{f(x^*+tv)-f(x^*)}{t} = v^{\mathrm{T}}\nabla f(x^*) + o(1).$$

根据 x^* 的最优性，在上式中分别对 t 取点 0 处的左、右极限可知

$$\lim_{t\to 0^+}\frac{f(x^*+tv)-f(x^*)}{t}=v^{\mathrm{T}}\nabla f(x^*)\geqslant 0,$$
$$\lim_{t\to 0^-}\frac{f(x^*+tv)-f(x^*)}{t}=v^{\mathrm{T}}\nabla f(x^*)\leqslant 0,$$

即对任意的 v 有 $v^{\mathrm{T}}\nabla f(x^*)=0$，由 v 的任意性知 $\nabla f(x^*)=0$. □

注意，上面的条件仅仅是必要的. 对于 $f(x)=x^2, x\in\mathbb{R}$，我们知道满足 $f'(x)=0$ 的点为 $x^*=0$，并且其也是全局最优解. 对于 $f(x)=x^3, x\in\mathbb{R}$，满足 $f'(x)=0$ 的点为 $x^*=0$，但其不是一个局部最优解. 实际上，我们称满足 $\nabla f(x)=0$ 的点 x 为 f 的**稳定点**（有时也称为驻点或临界点）. 可以看出，除了一阶必要条件，还需要对函数加一些额外的限制条件，才能保证最优解的充分性. 我们会在后面的小节中继续讨论.

4.2.2 二阶最优性条件

在没有额外假设时，如果一阶必要条件满足，我们仍然不能确定当前点是否是一个局部极小点. 这里考虑使用二阶信息来进一步判断给定点的最优性.

假设 f 在点 x 的一个开邻域内是二阶连续可微的. 类似于一阶必要条件的推导，可以借助当前点处的二阶泰勒展开来逼近该函数在该点附近的取值情况，从而来判断最优性. 具体地，在点 x 附近我们考虑泰勒展开

$$f(x+d)=f(x)+\nabla f(x)^{\mathrm{T}}d+\frac{1}{2}d^{\mathrm{T}}\nabla^2 f(x)d+o(\|d\|^2).$$

当一阶必要条件满足时，$\nabla f(x)=0$，那么上面的展开式简化为

$$f(x+d)=f(x)+\frac{1}{2}d^{\mathrm{T}}\nabla^2 f(x)d+o(\|d\|^2). \tag{4.2.2}$$

因此，我们有如下二阶最优性条件：

定理 4.3 假设 f 在点 x^* 的一个开邻域内是二阶连续可微的，则以下最优性条件成立：

二阶必要条件 如果 x^* 是 f 的一个局部极小点，那么

$$\nabla f(x^*)=0,\quad \nabla^2 f(x^*)\succeq 0;$$

二阶充分条件 如果在点 x^* 处有

$$\nabla f(x^*) = 0, \quad \nabla^2 f(x^*) \succ 0$$

成立，那么 x^* 为 f 的一个局部极小点.

证明. 考虑 $f(x)$ 在点 x^* 处的二阶泰勒展开(4.2.2)，这里因为一阶必要条件成立，所以 $\nabla f(x^*) = 0$. 反设 $\nabla^2 f(x^*) \succeq 0$ 不成立，即 $\nabla^2 f(x^*)$ 有负的特征值. 取 d 为其负特征值 λ_- 对应的特征向量，通过对(4.2.2)式变形得到

$$\frac{f(x^* + d) - f(x^*)}{\|d\|^2} = \frac{1}{2} \frac{d^{\mathrm{T}}}{\|d\|} \nabla^2 f(x^*) \frac{d}{\|d\|} + o(1).$$

这里注意 $\dfrac{d}{\|d\|}$ 是 d 的单位化，因此

$$\frac{f(x^* + d) - f(x^*)}{\|d\|^2} = \frac{1}{2} \lambda_- + o(1).$$

当 $\|d\|$ 充分小时，$f(x^* + d) < f(x^*)$，这和点 x^* 的最优性矛盾. 因此二阶必要条件成立.

当 $\nabla^2 f(x) \succ 0$ 时，对任意的 $d \neq 0$ 有 $d^{\mathrm{T}} \nabla^2 f(x^*) d \geqslant \lambda_{\min} \|d\|^2 > 0$，这里 $\lambda_{\min} > 0$ 是 $\nabla^2 f(x^*)$ 的最小特征值. 因此我们有

$$\frac{f(x^* + d) - f(x^*)}{\|d\|^2} \geqslant \frac{1}{2} \lambda_{\min} + o(1).$$

当 $\|d\|$ 充分小时有 $f(x^* + d) \geqslant f(x^*)$，即二阶充分条件成立. □

由定理4.3有如下结论：设点 $\bar{x}$ 满足一阶最优性条件（即 $\nabla f(\bar{x}) = 0$），且该点处的海瑟矩阵 $\nabla^2 f(\bar{x})$ 不是半正定的，那么 $\bar{x}$ 不是一个局部极小点. 进一步地，如果海瑟矩阵 $\nabla^2 f(\bar{x})$ 既有正特征值又有负特征值，我们称 稳定点 $\bar{x}$ 为一个**鞍点**. 事实上，记 d_1, d_2 为其正负特征值对应的特征向量，那么对于任意充分小的 $t > 0$，我们都有 $f(\bar{x} + t d_1) > f(\bar{x})$ 且 $f(\bar{x} + t d_2) < f(\bar{x})$.

注意，二阶最优性条件给出的仍然是关于局部最优性的判断. 对于给定点的全局最优性判断，我们还需要借助实际问题的性质，比如目标函数是凸的、非线性最小二乘问题中目标函数值为 0 等.

4.2.3 实例

我们以线性最小二乘问题为例来说明其最优性条件的具体形式. 如第3.2节中所述，线性最小二乘问题可以表示为

$$\min_{x\in\mathbb{R}^n} \quad f(x) \xlongequal{\text{def}} \frac{1}{2}\|b-Ax\|_2^2,$$

其中 $A\in\mathbb{R}^{m\times n}, b\in\mathbb{R}^m$ 分别是给定的矩阵和向量. 易知 $f(x)$ 是可微且凸的，因此，x^* 为一个全局最优解当且仅当

$$\nabla f(x^*) = A^{\mathrm{T}}(Ax^*-b) = 0.$$

因此，线性最小二乘问题本质上等于求解线性方程组，可以利用数值代数知识对其有效求解.

在实际中，我们还经常遇到非线性最小二乘问题，如实数情形的相位恢复问题，其一般形式如下：

$$\min_{x\in\mathbb{R}^n} \quad f(x) \xlongequal{\text{def}} \sum_{i=1}^m r_i^2(x), \tag{4.2.3}$$

其中非线性函数 $r_i(x) = (a_i^{\mathrm{T}}x)^2 - b_i^2, i=1,2,\cdots,m$. 这个问题是非凸的. 在点 $x\in\mathbb{R}^n$ 处，我们有

$$\begin{aligned}
\nabla f(x) &= 2\sum_{i=1}^m r_i(x)\nabla r_i(x) = 4\sum_{i=1}^m ((a_i^{\mathrm{T}}x)^2 - b_i^2)(a_i^{\mathrm{T}}x)a_i,\\
\nabla^2 f(x) &= 2\sum_{i=1}^m \nabla r_i(x)\nabla r_i(x)^{\mathrm{T}} + 2\sum_{i=1}^m r_i(x)\nabla^2 r_i(x)\\
&= 8\sum_{i=1}^m (a_i^{\mathrm{T}}x)^2 a_i a_i^{\mathrm{T}} + 4\sum_{i=1}^m ((a_i^{\mathrm{T}}x)^2 - b_i^2)a_i a_i^{\mathrm{T}}\\
&= \sum_{i=1}^m (12(a_i^{\mathrm{T}}x)^2 - 4b_i^2)a_i a_i^{\mathrm{T}}.
\end{aligned}$$

如果 x^* 为问题 (4.2.3) 的一个局部最优解，那么其满足一阶必要条件

$$\nabla f(x^*) = 0,$$

即

$$\sum_{i=1}^m ((a_i^{\mathrm{T}}x^*)^2 - b_i)(a_i^{\mathrm{T}}x^*)a_i = 0,$$

以及二阶必要条件

$$\nabla^2 f(x^*) \succeq 0,$$

即

$$\sum_{i=1}^{m}(12(a_i^{\mathrm{T}}x^*)^2-4b_i^2)a_ia_i^{\mathrm{T}}\succeq 0.$$

如果一个点 $x^{\#}$ 满足二阶充分条件

$$\nabla f(x^{\#})=0,\quad \nabla^2 f(x^{\#})\succ 0,$$

即

$$\sum_{i=1}^{m}((a_i^{\mathrm{T}}x^{\#})^2-b_i)(a_i^{\mathrm{T}}x^{\#})a_i=0,\quad \sum_{i=1}^{m}(12(a_i^{\mathrm{T}}x^{\#})^2-4b_i^2)a_ia_i^{\mathrm{T}}\succ 0,$$

那么 $x^{\#}$ 为问题 (4.2.3) 的一个局部最优解.

4.3 无约束不可微问题的最优性理论

本节仍考虑问题 (4.2.1):

$$\min_{x\in\mathbb{R}^n}\quad f(x),$$

但其中 $f(x)$ 为不可微函数. 很多实际问题的目标函数不是光滑的，例如 $f(x)=\|x\|_1$. 对于此类问题，由于目标函数可能不存在梯度和海瑟矩阵，因此第4.2节中的一阶和二阶条件不适用. 此时我们必须使用其他最优性条件来判断不可微问题的最优点.

4.3.1 凸优化问题一阶充要条件

对于目标函数是凸函数的情形，我们已经引入了次梯度的概念并给出了其计算法则. 一个自然的问题是：可以利用次梯度代替梯度来构造最优性条件吗？答案是肯定的，实际上有如下定理：

定理 4.4 *假设 f 是适当且凸的函数，则 x^* 为问题 (4.2.1) 的一个全局极小点当且仅当*

$$0\in\partial f(x^*).$$

证明. 先证必要性. 因为 x^* 为全局极小点，所以

$$f(y)\geqslant f(x^*)=f(x^*)+0^{\mathrm{T}}(y-x^*),\quad \forall y\in\mathbb{R}^n.$$

因此，$0\in\partial f(x^*)$.

再证充分性. 如果 $0 \in \partial f(x^*)$，那么根据次梯度的定义

$$f(y) \geqslant f(x^*) + 0^{\mathrm{T}}(y - x^*) = f(x^*), \quad \forall y \in \mathbb{R}^n.$$

因而 x^* 为一个全局极小点. □

定理4.4说明条件 $0 \in \partial f(x^*)$ 是 x^* 为全局最优解的充要条件. 这个结论比定理4.2要强，其原因是凸问题有非常好的性质，它的稳定点中不存在鞍点. 因此，可以通过计算凸函数的次梯度集合来求解其对应的全局极小点. 相较于非凸函数，凸函数的最优性分析简单，计算以及验证起来比较方便，因此在实际建模中受到广泛的关注.

4.3.2 复合优化问题的一阶必要条件

在实际问题中，目标函数不一定是凸函数，但它可以写成一个光滑函数与一个非光滑凸函数的和，例如第3.3节介绍的复合优化问题就具有这样的形式. 其中目标函数的光滑项可能是凸的，比如 LASSO 问题、图像去噪问题和盲反卷积问题；也可能是非凸的，例如字典学习问题和神经网络的损失函数. 因此研究此类问题的最优性条件十分必要. 这里，我们考虑一般复合优化问题

$$\min_{x \in \mathbb{R}^n} \quad \psi(x) \xlongequal{\text{def}} f(x) + h(x), \tag{4.3.1}$$

其中 f 为光滑函数（可能非凸），h 为凸函数（可能非光滑）. 对于其任何局部最优解，我们不加证明地给出如下一阶必要条件：

定理 4.5 (复合优化问题一阶必要条件) *令 x^* 为问题 (4.3.1) 的一个局部极小点，那么*

$$-\nabla f(x^*) \in \partial h(x^*),$$

其中 $\partial h(x^*)$ 为凸函数 h 在点 x^* 处的次梯度集合.

定理4.5在之后我们推导复合优化问题算法性质的时候非常重要，它给出了当目标函数一部分是非光滑凸函数时的一阶必要条件. 在这里注意，由于目标函数可能是整体非凸的，因此一般没有一阶充分条件. 在第七章中我们介绍邻近算子时会用到这个定理.

4.3.3 实例

我们以 ℓ_1 范数优化问题为例，给出其最优解的最优性条件. 第一章和第三章介绍了各种各样的 ℓ_1 范数优化问题，其一般形式可以写成

$$\min_{x\in\mathbb{R}^n} \quad \psi(x) \overset{\text{def}}{=\!=} f(x) + \mu\|x\|_1, \tag{4.3.2}$$

其中 $f(x):\mathbb{R}^n \to \mathbb{R}$ 为光滑函数，正则系数 $\mu > 0$ 用来调节解的稀疏度. 尽管 $\|x\|_1$ 不是可微的，但我们可以计算其次微分

$$\partial_i \|x\|_1 = \begin{cases} \{1\}, & x_i > 0, \\ [-1,1], & x_i = 0, \\ \{-1\}, & x_i < 0. \end{cases}$$

因此，如果 x^* 是问题 (4.3.2) 的一个局部最优解，那么其满足

$$-\nabla f(x^*) \in \mu\partial\|x^*\|_1,$$

即

$$\nabla_i f(x^*) = \begin{cases} -\mu, & x_i^* > 0, \\ a \in [-\mu,\mu], & x_i^* = 0, \\ \mu, & x_i^* < 0. \end{cases}$$

进一步地，如果 $f(x)$ 是凸的（比如在 LASSO 问题中 $f(x) = \frac{1}{2}\|Ax - b\|^2$），那么满足上式的 x^* 就是问题 (4.3.2) 的全局最优解.

4.4 对偶理论

这一节以及本章之后的章节考虑一般的约束优化问题

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x), \\ \text{s.t.} \quad & c_i(x) \leqslant 0,\ i \in \mathcal{I}, \\ & c_i(x) = 0,\ i \in \mathcal{E}, \end{aligned} \tag{4.4.1}$$

其中 c_i 为定义在 $\mathbb{R}^n$ 或其子集上的实值函数，$\mathcal{I}$ 和 $\mathcal{E}$ 分别表示不等式约束和等式约束对应的下标集合且各下标互不相同. 这个问题的可行域定义为

$$\mathcal{X} = \{x \in \mathbb{R}^n | c_i(x) \leqslant 0,\ i \in \mathcal{I} \text{ 且 } c_i(x) = 0,\ i \in \mathcal{E}\}.$$

我们可以通过将 $\mathcal{X}$ 的示性函数加到目标函数中得到无约束优化问题. 但是转化后问题的目标函数是不连续的、不可微的以及不是有限的，这导致我们难以分析其理论性质以及设计有效的算法. 对于约束优化问题，可行性问题是应该最先考虑的. 因此，对其约束集合的几何性质以及代数性质的分析尤为重要.

4.4.1 拉格朗日函数与对偶问题

研究问题 (4.4.1) 的重要工具之一是拉格朗日函数，它的基本思想是给该问题中的每一个约束指定一个**拉格朗日乘子**，以乘子为加权系数将约束增加到目标函数中. 令 λ_i 为对应于第 i 个不等式约束的拉格朗日乘子，ν_i 为对应于第 i 个等式约束的拉格朗日乘子. 为了构造合适的对偶问题，基本原则是对拉格朗日乘子添加合适的约束条件，使得 $f(x)$ 在问题 (4.4.1) 的任意可行点 x 处大于或等于相应拉格朗日函数值. 根据这个原则，我们要求 $\lambda \geqslant 0$. 记 $m = |\mathcal{I}|, p = |\mathcal{E}|$，则**拉格朗日函数**的具体形式 $L : \mathbb{R}^n \times \mathbb{R}^m_+ \times \mathbb{R}^p \to \mathbb{R}$ 定义为

$$L(x,\lambda,\nu) = f(x) + \sum_{i\in\mathcal{I}} \lambda_i c_i(x) + \sum_{i\in\mathcal{E}} \nu_i c_i(x). \tag{4.4.2}$$

注意，函数 (4.4.2) 中的加号也可以修改为减号，同时调整相应乘子的约束条件使得上述下界原则满足即可.

对拉格朗日函数 $L(x,\lambda,\nu)$ 中的 x 取下确界可定义**拉格朗日对偶函数**，这一函数将在对偶理论中起到很关键的作用.

定义 4.2 拉格朗日对偶函数 $g : \mathbb{R}^m_+ \times \mathbb{R}^p \to [-\infty, +\infty)$ 是拉格朗日函数 $L(x,\lambda,\nu)$ 对于 $\lambda \in \mathbb{R}^m_+, \nu \in \mathbb{R}^p$ 关于 x 取的下确界：

$$g(\lambda,\nu) = \inf_{x\in\mathbb{R}^n} L(x,\lambda,\nu). \tag{4.4.3}$$

固定 (λ,ν)，如果拉格朗日函数关于 x 无界，那么对偶函数在 (λ,ν) 处的取值为 $-\infty$. 因为拉格朗日对偶函数是逐点定义的一族关于 (λ,ν) 的仿射函数的下确界，根据定理 2.11的(5)可知其为凹函数（无论原始问题是否为凸问题）. 这个性质是十分重要的，它能帮助我们推导出许多拥有良好性质的算法.

对每一对满足 $\lambda \geqslant 0$ 的乘子对 (λ,ν)，拉格朗日对偶函数 $g(\lambda,\nu)$ 给原优化问题(4.4.1)的最优值 p^* 提供了下界，且该下界依赖于参数 λ 和 ν 的选取.

引理 4.1 (弱对偶原理) 对于任意的 $\lambda \geqslant 0$ 和 ν，拉格朗日对偶函数给出了优化问题(4.4.1)最优值的一个下界，即

$$g(\lambda,\nu) \leqslant p^*, \quad \lambda \geqslant 0. \tag{4.4.4}$$

证明. 假设 $\tilde{x}$ 是问题(4.4.1)的一个可行解，即 $c_i(\tilde{x}) \leqslant 0, i \in \mathcal{I}$ 和 $c_i(\tilde{x}) = 0, i \in \mathcal{E}$ 对于任意的 i 均成立. 由于 $\lambda \geqslant 0$，则

$$\sum_{i\in\mathcal{I}} \lambda_i c_i(\tilde{x}) + \sum_{i\in\mathcal{E}} \nu_i c_i(\tilde{x}) \leqslant 0. \tag{4.4.5}$$

将上式代入拉格朗日函数的定义中，我们可以得到

$$L(\tilde{x},\lambda,\nu) = f(\tilde{x}) + \sum_{i\in\mathcal{I}} \lambda_i c_i(\tilde{x}) + \sum_{i\in\mathcal{E}} \nu_i c_i(\tilde{x}) \leqslant f(\tilde{x}), \tag{4.4.6}$$

并且

$$g(\lambda,\nu) = \inf_x L(x,\lambda,\nu) \leqslant L(\tilde{x},\lambda,\nu) \leqslant f(\tilde{x}). \tag{4.4.7}$$

所以对于任意的可行解 $\tilde{x}$, $g(\lambda,\nu) \leqslant f(\tilde{x})$ 都成立，从而 $g(\lambda,\nu) \leqslant p^*$ 成立. □

那么一个自然的问题是，从拉格朗日对偶函数获得的下界中，哪个是最优的呢？为了求解该最优的下界，便有如下**拉格朗日对偶问题**：

$$\max_{\lambda\geqslant 0,\nu} g(\lambda,\nu) = \max_{\lambda\geqslant 0,\nu} \inf_{x\in\mathbb{R}^n} L(x,\lambda,\nu). \tag{4.4.8}$$

向量 λ 和 ν 也称为问题(4.4.1) 的**对偶变量**或者拉格朗日乘子向量. 由于其目标函数的凹性和约束集合的凸性，拉格朗日对偶问题是一个凸优化问题（见注 1.1）. 当 $g(\lambda,\nu) = -\infty$ 时，对偶函数提供的 p^* 的下界变得没有实际意义. 只有当 $g(\lambda,\nu) > -\infty$ 时，对偶函数生成的关于原始问题最优解 p^* 的下界才是非平凡的. 因此我们规定拉格朗日对偶函数的定义域

$$\mathbf{dom}\, g = \{(\lambda,\nu) \mid \lambda \geqslant 0, g(\lambda,\nu) > -\infty\}.$$

当 $(\lambda,\nu) \in \mathbf{dom}\, g$ 时，称其为**对偶可行解**. 记对偶问题的最优值为 q^*. 称 $p^* - q^* (\geqslant 0)$ 为**对偶间隙**. 如果对偶间隙为 $0 (p^* = q^*)$，称**强对偶原理**成立. 假设 (λ^*,ν^*) 是使得对偶问题取得最优值的解，称其为**对偶最优解**或者最优拉格朗日乘子.

推导拉格朗日对偶问题最重要的是能把拉格朗日对偶函数的具体形式方便地写出来. 需要指出的是，拉格朗日对偶问题的写法并不唯一. 如果问

题 (4.4.1) 中有些约束，比如对应于下标集 $\mathcal{I}_1 = \{i_1, i_2, \cdots, i_q\}$ 的不等式约束，比较简单，则可以不把这些约束松弛到拉格朗日函数里. 此时拉格朗日函数为

$$L(x,s,\nu) = f(x) + \sum_{i\in\mathcal{I}\setminus\mathcal{I}_1} s_i c_i(x) + \sum_{i\in\mathcal{E}} \nu_i c_i(x),\ s \geqslant 0,\ c_i(x) \leqslant 0,\ i \in \mathcal{I}_1, \quad (4.4.9)$$

相应地，对偶问题为

$$\max_{s\geqslant 0,\nu} \left\{ \inf_{x\in\mathbb{R}^n} L(x,s,\nu),\ \text{s.t. } c_i(x) \leqslant 0, i \in \mathcal{I}_1 \right\}. \quad (4.4.10)$$

对于强对偶原理满足的凸问题，不同写法的拉格朗日对偶问题是等价的.

4.4.2 带广义不等式约束优化问题的对偶

问题 (4.4.1) 中的不等式约束 $c_i(x),\ i \in \mathcal{I}$ 都是实值函数的形式. 在许多实际应用中，我们还会遇到大量带广义不等式约束的优化问题，例如自变量 x 可能取值于半正定矩阵空间中. 对于这类约束我们不易将其化为 $c_i(x) \leqslant 0$ 的形式，此时又该如何构造拉格朗日对偶函数呢?

1. 适当锥和广义不等式

定义广义不等式需要利用适当锥的概念.

定义 4.3 (适当锥) 称满足如下条件的锥 K 为**适当锥** (proper cone):

(1) K 是凸锥;

(2) K 是闭集;

(3) K 是实心的 (solid), 即 $\mathbf{int}K \neq \varnothing$;

(4) K 是尖的 (pointed), 即对任意非零向量 x, 若 $x \in K$, 则 $-x \notin K$, 也即 K 中无法容纳直线.

适当锥 K 可以诱导出广义不等式，它定义了全空间上的偏序关系:

$$x \preceq_K y \iff y - x \in K.$$

类似地，可以定义严格广义不等式:

$$x \prec_K y \iff y - x \in \mathbf{int}K.$$

本书常用的是 $K=\mathbb{R}^n_+$（非负锥）和 $K=\mathcal{S}^n_+$（半正定锥）. 当 $K=\mathbb{R}^n_+$ 时，$x\preceq_K y$ 是我们之前经常使用的记号 $x\leqslant y$，即 x 每个分量小于等于 y 的对应分量；当 $K=\mathcal{S}^n_+$ 时，$X\preceq_K Y$ 的含义为 $Y-X\succeq 0$，即 $Y-X$ 是半正定矩阵.

广义不等式满足许多我们熟悉的性质，例如自反性、反对称性、传递性，这里不详细展开.

2. 对偶锥

在构造拉格朗日对偶函数时，针对不等式约束 $c_i(x)\leqslant 0$ 我们引入拉格朗日乘子 $\lambda_i\geqslant 0$，之后将 $\lambda_i c_i(x)$（$\leqslant 0$）作为拉格朗日函数中的一项. 那么对于广义不等式，应该如何对拉格朗日乘子提出限制呢？此时需要借助**对偶锥**的概念.

定义 4.4 (对偶锥) 令 K 为全空间 Ω 的子集，称集合

$$K^*=\{y\in\Omega\mid\langle x,y\rangle\geqslant 0,\quad\forall x\in K\}$$

为其对偶锥，其中 $\langle\cdot,\cdot\rangle$ 是 Ω 上的一个内积.

正如其定义所说，对偶锥是一个锥（哪怕原始集合 K 不是锥）. 我们在图 4.1 中给出了 $\mathbb{R}^2$ 平面上的一个例子. 图中深色区域表示锥 K，根据对偶锥的定义，K^* 中的向量和 K 中所有向量夹角均为锐角或直角. 因此，对偶锥 K^* 为图 4.1 的浅色区域. 注意，在这个例子中 K 也为 K^* 一部分.

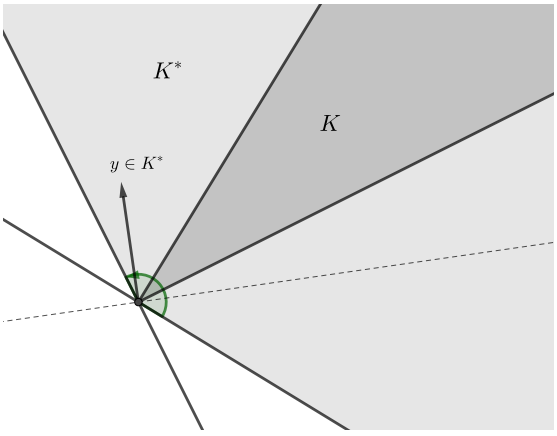


图 4.1 $\mathbb{R}^2$ 平面上的锥 K 及其对偶锥 K^*

如果 $K=\mathbb{R}^n_+,\Omega=\mathbb{R}^n$ 并且定义 $\langle x,y\rangle=x^{\mathrm{T}}y$，那么易知 $K^*=\mathbb{R}^n_+$. 假设 $K=\mathcal{S}^n_+,\Omega=\mathcal{S}^n$ 并且定义 $\langle X,Y\rangle=\mathrm{Tr}(XY^{\mathrm{T}})$，可以证明（见习题 4.3）

$$\langle X,Y\rangle\geqslant 0,\ \forall X\in\mathcal{S}^n_+ \quad\Longleftrightarrow\quad Y\in\mathcal{S}^n_+,$$

即半正定锥的对偶锥仍为半正定锥. 此外，称满足 $K=K^*$ 的锥 K 为**自对偶锥**，因此非负锥和半正定锥都是自对偶锥.

直观来说，对偶锥 K^* 中向量和原锥 K 中向量的内积恒非负，这一性质可以被用来构造拉格朗日对偶函数.

3. 广义不等式约束优化问题拉格朗日函数的构造

如果将不等式约束函数换成向量函数，并且推广定义相应的广义不等式约束，我们可以得到如下形式的优化问题：

$$\begin{aligned}\min_{x\in\mathbb{R}^n}\quad & f(x),\\ \text{s.t.}\quad & c_i(x)\preceq_{K_i}0,\ i\in\mathcal{I},\\ & c_i(x)=0,\ i\in\mathcal{E},\end{aligned}\tag{4.4.11}$$

其中 $f:\mathbb{R}^n\to\mathbb{R}$, $c_i:\mathbb{R}^n\to\mathbb{R}$, $i\in\mathcal{E}$ 为实值函数，$c_i:\mathbb{R}^n\to\mathbb{R}^{k_i}$, $k_i\in\mathbb{N}_+,i\in\mathcal{I}$ 为向量值函数，K_i 为某种适当锥且 $\preceq_{K_i}$ 表示由锥 K_i 定义的广义不等式. 因此，问题 (4.4.1) 是在问题 (4.4.11) 中取 $k_i=1$, $K_i=\mathbb{R}_+$, $\forall i\in\mathcal{I}$ 时的特殊情形.

根据 K_i, $i\in\mathcal{I}$ 的对偶锥 K_i^*，我们对广义不等式约束分别引入乘子 $\lambda_i\in K_i^*$, $i\in\mathcal{I}$，对等式约束引入乘子 $\nu_i\in\mathbb{R}$, $i\in\mathcal{E}$，构造如下拉格朗日函数：

$$L(x,\lambda,\nu)=f(x)+\sum_{i\in\mathcal{I}}\langle c_i(x),\lambda_i\rangle+\sum_{i\in\mathcal{E}}\nu_ic_i(x),\quad \lambda_i\in K_i^*,\nu_i\in\mathbb{R}.$$

容易验证 $L(x,\lambda,\nu)\leqslant f(x)$, $\forall\, x\in\mathcal{X}$, $\lambda_i\in K_i^*,\nu_i\in\mathbb{R}$. 式我们定义拉格朗日对偶函数

$$g(\lambda,\nu)=\inf_{x\in\mathbb{R}^n}L(x,\lambda,\nu).$$

因此，对偶问题为

$$\max_{\lambda_i\in K_i^*,\ \nu_i\in\mathbb{R}}\quad g(\lambda,\nu).$$

对偶问题在最优化理论中扮演着重要角色. 每个优化问题都对应一个对偶问题. 相比原始问题，对偶问题总是凸的，其最优值给出了原始问题最优

值（极小化问题）的一个下界. 如果原始问题满足一定的条件，我们可以从理论上证明原始问题和对偶问题的最优值是相等的. 当原始问题的约束个数比决策变量维数更小时，对偶问题的决策变量维数会比原始问题的小，从而可能在相对较小的决策空间中求解. 因此，对于对偶问题的研究非常必要. 接下来我们会给出一些例子来说明如何求出给定问题的对偶问题.

4.4.3 实例

这一小节用四个例子说明拉格朗日对偶问题应当如何计算，并简要从对偶理论的角度分析这些问题具有的性质.

1. 线性规划问题的对偶

考虑如下线性规划问题：

$$\begin{aligned} \min_{x} \quad & c^{\mathrm{T}}x, \\ \text{s.t.} \quad & Ax = b, \\ & x \geqslant 0. \end{aligned} \tag{4.4.12}$$

对于等式约束 $Ax = b$，我们引入拉格朗日乘子 ν；对于非负约束 $x \geqslant 0$，我们引入拉格朗日乘子 $s \geqslant 0$. 根据上述准则，可构造如下拉格朗日函数：

$$L(x,s,\nu) = c^{\mathrm{T}}x + \nu^{\mathrm{T}}(Ax - b) - s^{\mathrm{T}}x = -b^{\mathrm{T}}\nu + (A^{\mathrm{T}}\nu - s + c)^{\mathrm{T}}x,$$

其拉格朗日对偶函数为

$$g(s,\nu) = \inf_{x} L(x,s,\nu) = \begin{cases} -b^{\mathrm{T}}\nu, & A^{\mathrm{T}}\nu - s + c = 0, \\ -\infty, & \text{其他}. \end{cases}$$

注意到只需考虑 $A^{\mathrm{T}}\nu - s + c = 0$ 情形，其余情况对应于不可行情形，因此线性规划问题(4.4.12)的对偶问题是

$$\begin{aligned} \max_{s,\nu} \quad & -b^{\mathrm{T}}\nu, \\ \text{s.t.} \quad & A^{\mathrm{T}}\nu - s + c = 0, \\ & s \geqslant 0. \end{aligned}$$

经过变量代换 $y=-v$ 后，上述问题等价于常见的形式

$$\begin{aligned} \max_{s,y} \quad & b^{\mathrm{T}}y, \\ \text{s.t.} \quad & A^{\mathrm{T}}y+s=c, \\ & s\geqslant 0. \end{aligned} \tag{4.4.13}$$

如果问题(4.4.12)的拉格朗日函数直接写为

$$L(x,s,y)=c^{\mathrm{T}}x-y^{\mathrm{T}}(Ax-b)-s^{\mathrm{T}}x=b^{\mathrm{T}}y+(c-A^{\mathrm{T}}y-s)^{\mathrm{T}}x,$$

其中 y 为等式约束 $Ax=b$ 的乘子以及 $s\geqslant 0$ 为非负约束 $x\geqslant 0$ 的乘子，则对偶问题(4.4.13)可以直接导出.

线性规划问题(4.4.12)的对偶问题也可以通过保留约束 $x\geqslant 0$ 写出. 对于等式约束 $Ax=b$，引入乘子 y，则相应的拉格朗日函数为

$$L(x,y)=c^{\mathrm{T}}x-y^{\mathrm{T}}(Ax-b)=b^{\mathrm{T}}y+(c-A^{\mathrm{T}}y)^{\mathrm{T}}x.$$

而对偶问题需要将 $x\geqslant 0$ 添加到约束里：

$$\max_{y}\left\{\inf_{x} b^{\mathrm{T}}y+(c-A^{\mathrm{T}}y)^{\mathrm{T}}x, \quad \text{s.t.} \quad x\geqslant 0\right\}.$$

简化后得出：

$$\begin{aligned} \max_{y} \quad & b^{\mathrm{T}}y, \\ \text{s.t.} \quad & A^{\mathrm{T}}y\leqslant c. \end{aligned} \tag{4.4.14}$$

事实上，由对偶问题(4.4.13)也可以消掉变量 s 得到(4.4.14).

下面我们推导问题(4.4.14)的对偶问题. 先通过极小化目标函数的相反数将其等价地转化为如下极小化问题（另一种方式是直接构造极大化问题的拉格朗日函数，通过引入乘子并确定其符号使得构造的拉格朗日函数为 $b^{\mathrm{T}}y$ 的一个上界，之后再求拉格朗日函数关于 x 的上确界得对偶函数）：

$$\begin{aligned} \min_{y} \quad & -b^{\mathrm{T}}y, \\ \text{s.t.} \quad & A^{\mathrm{T}}y\leqslant c. \end{aligned}$$

对于不等式约束 $A^{\mathrm{T}}y\leqslant c$，我们引入拉格朗日乘子 $x\geqslant 0$，则相应的拉格朗日函数为

$$L(y,x)=-b^{\mathrm{T}}y+x^{\mathrm{T}}(A^{\mathrm{T}}y-c)=-c^{\mathrm{T}}x+(Ax-b)^{\mathrm{T}}y.$$

因此得到对偶函数

$$g(x)=\inf_y L(y,x)=\begin{cases}-c^{\mathrm{T}}x, & Ax=b,\\ -\infty, & \text{其他},\end{cases}$$

相应的对偶问题是

$$\begin{aligned}\max_x \quad & -c^{\mathrm{T}}x,\\ \text{s.t.} \quad & Ax=b,\\ & x\geqslant 0.\end{aligned}\tag{4.4.15}$$

观察到问题 (4.4.15) 与问题 (4.4.12) 完全等价，这表明线性规划问题与其对偶问题互为对偶.

2. ℓ_1 正则化问题的对偶

对于 ℓ_1 正则化问题

$$\min_{x\in\mathbb{R}^n} \quad \frac{1}{2}\|Ax-b\|^2+\mu\|x\|_1,\tag{4.4.16}$$

其中 $A\in\mathbb{R}^{m\times n}, b\in\mathbb{R}^m$ 分别为给定的矩阵和向量，μ 为正则化参数来控制稀疏度. 通过引入 $Ax-b=r$，可以将问题 (4.4.16) 转化为如下等价的形式：

$$\min_{x\in\mathbb{R}^n} \quad \frac{1}{2}\|r\|^2+\mu\|x\|_1, \quad \text{s.t.} \quad Ax-b=r,\tag{4.4.17}$$

其拉格朗日函数为

$$\begin{aligned}L(x,r,\lambda)&=\frac{1}{2}\|r\|^2+\mu\|x\|_1-\langle\lambda,Ax-b-r\rangle\\ &=\frac{1}{2}\|r\|^2+\lambda^{\mathrm{T}}r+\mu\|x\|_1-(A^{\mathrm{T}}\lambda)^{\mathrm{T}}x+b^{\mathrm{T}}\lambda.\end{aligned}$$

利用二次函数最小值的性质及 $\|\cdot\|_1$ 的对偶范数的定义，我们有

$$g(\lambda)=\inf_{x,r} L(x,r,\lambda)=\begin{cases}b^{\mathrm{T}}\lambda-\frac{1}{2}\|\lambda\|^2, & \|A^{\mathrm{T}}\lambda\|_\infty\leqslant\mu,\\ -\infty, & \text{其他}.\end{cases}$$

那么对偶问题为

$$\max \quad b^{\mathrm{T}}\lambda-\frac{1}{2}\|\lambda\|^2, \quad \text{s.t.} \quad \|A^{\mathrm{T}}\lambda\|_\infty\leqslant\mu.$$

3. 半定规划问题的对偶问题

考虑标准形式的半定规划问题

$$\begin{aligned} \min_{X\in\mathcal{S}^n} \quad & \langle C, X\rangle, \\ \text{s.t.} \quad & \langle A_i, X\rangle = b_i,\ i=1,2,\cdots,m, \\ & X \succeq 0, \end{aligned} \tag{4.4.18}$$

其中 $A_i \in \mathcal{S}^n$, $i=1,2,\cdots,m$, $C\in\mathcal{S}^n$,$b\in\mathbb{R}^m$. 对于等式约束和半正定锥约束分别引入乘子 $y\in\mathbb{R}^m$ 和 $S\in\mathcal{S}^n_+$，拉格朗日函数可以写为

$$L(X,y,S) = \langle C,X\rangle - \sum_{i=1}^m y_i(\langle A_i,X\rangle - b_i) - \langle S,X\rangle, \quad S\succeq 0,$$

则对偶函数为

$$g(y,S) = \inf_X L(X,y,S) = \begin{cases} b^{\mathrm{T}}y, & \sum_{i=1}^m y_iA_i - C + S = 0, \\ -\infty, & \text{其他}. \end{cases}$$

因此，对偶问题为

$$\begin{aligned} \min_{y\in\mathbb{R}^m} \quad & -b^{\mathrm{T}}y, \\ \text{s.t.} \quad & \sum_{i=1}^m y_iA_i - C + S = 0, \\ & S\succeq 0. \end{aligned} \tag{4.4.19}$$

它也可以写成不等式形式

$$\begin{aligned} \min_{y\in\mathbb{R}^m} \quad & -b^{\mathrm{T}}y, \\ \text{s.t.} \quad & \sum_{i=1}^m y_iA_i \preceq C. \end{aligned} \tag{4.4.20}$$

对于对偶问题(4.4.20)，我们还可以求其对偶问题. 对不等式约束引入乘子 $X\in\mathcal{S}^n$ 并且 $X\succeq 0$，拉格朗日函数为

$$\begin{aligned} L(y,X) &= -b^{\mathrm{T}}y + \langle X, \left(\sum_{i=1}^m y_iA_i\right) - C\rangle, \\ &= \sum_{i=1}^m y_i(-b_i + \langle A_i,X\rangle) - \langle C,X\rangle. \end{aligned}$$

因为上式对 y 是仿射的，故对偶函数可以描述为

$$g(X)=\inf_y L(y,X)=\begin{cases}-\langle C,X\rangle, & \langle A_i,X\rangle=b_i,\ i=1,2,\cdots,m,\\ -\infty, & \text{其他.}\end{cases}$$

因此，对偶问题可以写成

$$\begin{aligned}\min_{X\in\mathcal{S}^n}\quad & \langle C,X\rangle,\\ \text{s.t.}\quad & \langle A_i,X\rangle=b_i,\ i=1,2,\cdots,m,\\ & X\succeq 0.\end{aligned}\tag{4.4.21}$$

这就是问题 (4.4.18)，即半定规划问题与其对偶问题互为对偶.

4.5 一般约束优化问题的最优性理论

4.5.1 一阶最优性条件

类似于无约束优化问题，约束优化问题 (4.4.1) 的最优性条件要从下降方向开始讨论. 因为决策变量限制在可行域当中，所以只需要关注“可行”的方向. 先引入可行域的几何性质.

1. 切锥和约束品性

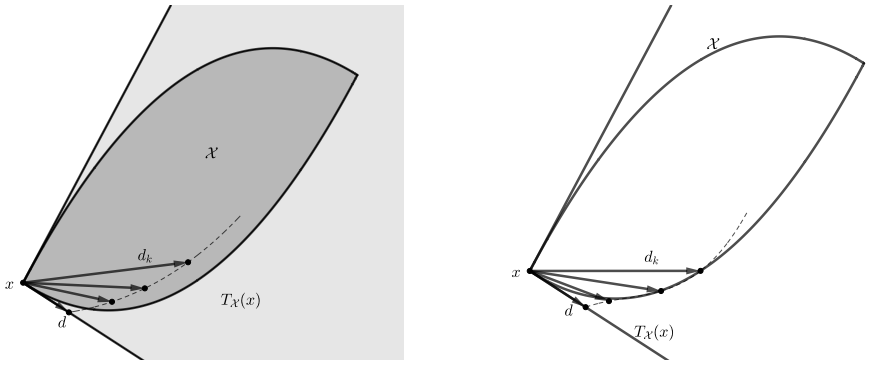
在给出最优性条件之前，我们先介绍一些必要的概念. 与无约束优化问题类似，首先需要定义问题 (4.4.1) 的下降方向. 这里因为约束的存在，我们只考虑可行方向，即可行序列对应的极限方向. 特别地，称这样的方向为切向量.

定义 4.5 (切锥) 给定可行域 $\mathcal{X}$ 及其内一点 x, 若存在可行序列 $\{z_k\}_{k=1}^{\infty}\subset\mathcal{X}$ 逼近 x（即 $\lim\limits_{k\to\infty}z_k=x$）以及正标量序列 $\{t_k\}_{k=1}^{\infty}$, $t_k\to 0$ 满足

$$\lim_{k\to\infty}\frac{z_k-x}{t_k}=d,$$

则称向量 d 为 $\mathcal{X}$ 在点 x 处的一个切向量. 点 x 处的所有切向量构成的集合称为**切锥**，用 $T_{\mathcal{X}}(x)$ 表示.

我们以 $\mathbb{R}^2$ 上的约束集合为例来直观地给出切锥的几何结构. 图 4.2(a) 中深色区域 $\mathcal{X}$ 表示两个不等式约束，其在点 x 处的切锥 $T_{\mathcal{X}}(x)$ 为图中浅色区域. 注意，$\mathcal{X}$ 也为 $T_{\mathcal{X}}(x)$ 的一部分. 图 4.2(b) 中则是考虑等式约束，这里可行域 $\mathcal{X}$ 是图 4.2(a) 中可行域的边界. 根据切锥的定义，此时 $T_{\mathcal{X}}(x)$ 对应点 x 处与 $\mathcal{X}$ 相切的两条射线.



(a) 不等式约束　　(b) 等式约束

图 4.2　$\mathbb{R}^2$ 上的约束和切锥

有了切锥的定义之后，可以从几何上刻画问题 (4.4.1) 的最优性条件. 与无约束优化类似，我们要求切锥（可行方向集合）不包含使得目标函数值下降的方向. 具体地，有下面的一阶必要条件，称为几何最优性条件.

定理 4.6 (几何最优性条件[120]) *假设可行点 x^* 是问题 (4.4.1) 的一个局部极小点. 如果 $f(x)$ 和 $c_i(x), i \in \mathcal{I} \cup \mathcal{E}$ 在点 x^* 处是可微的，那么*

$$d^{\mathrm{T}} \nabla f(x^*) \geqslant 0, \quad \forall d \in T_{\mathcal{X}}(x^*)$$

等价于

$$T_{\mathcal{X}}(x^*) \cap \{d \mid \nabla f(x^*)^{\mathrm{T}} d < 0\} = \varnothing.$$

证明. 采用反证法. 假设在点 x^* 处有 $T_{\mathcal{X}}(x^*) \cap \{d \mid \nabla f(x^*)^{\mathrm{T}} d < 0\} \neq \varnothing$，令 $d \in T_{\mathcal{X}}(x^*) \cap \{d \mid \nabla f(x^*)^{\mathrm{T}} d < 0\}$. 根据切向量的定义，存在 $\{t_k\}_{k=1}^{\infty}$ 和 $\{d_k\}_{k=1}^{\infty}$ 使得 $x^* + t_k d_k \in \mathcal{X}$，其中 $t_k \to 0$ 且 $d_k \to d$. 由于 $\nabla f(x^*)^{\mathrm{T}} d < 0$，对

于充分大的 k，我们有

$$\begin{aligned} f(x^* + t_k d_k) &= f(x^*) + t_k \nabla f(x^*)^{\mathrm{T}} d_k + o(t_k) \\ &= f(x^*) + t_k \nabla f(x^*)^{\mathrm{T}} d + t_k \nabla f(x^*)^{\mathrm{T}} (d_k - d) + o(t_k) \\ &= f(x^*) + t_k \nabla f(x^*)^{\mathrm{T}} d + o(t_k) \\ &< f(x^*). \end{aligned}$$

这与 x^* 的局部极小性矛盾. □

因为切锥是根据可行域的几何性质来定义的，其计算往往是不容易的. 因此，我们需要寻找代数方法来计算可行方向，进而更容易地判断最优性条件. 我们给出另一个容易计算的可行方向集合的定义，即线性化可行方向锥.

定义 4.6 (线性化可行方向锥) 对于可行点 $x \in \mathcal{X}$，该点处的积极集 (active set) $\mathcal{A}(x)$ 定义为两部分下标的集合，一部分是等式约束对应的下标，另外一部分是不等式约束中等号成立的约束对应的下标，即

$$\mathcal{A}(x) = \mathcal{E} \cup \{i \in \mathcal{I} : c_i(x) = 0\}.$$

进一步地，点 x 处的**线性化可行方向锥**定义为

$$\mathcal{F}(x) = \left\{ d \;\middle|\; \begin{array}{l} d^{\mathrm{T}} \nabla c_i(x) = 0, \forall\, i \in \mathcal{E}, \\ d^{\mathrm{T}} \nabla c_i(x) \leqslant 0, \forall\, i \in \mathcal{A}(x) \cap \mathcal{I} \end{array} \right\}$$

图 4.3 直观地展示了 $\mathbb{R}^2$ 中的不等式约束集合和线性化可行方向锥 $\mathcal{F}(x)$. 在点 x 处，已知两个不等式约束的等号均成立. 而 $\mathcal{F}(x)$ 中的向量应保证和 $\nabla c_i(x)$, $i = 1,2$ 的夹角为钝角或直角.

直观来说，线性化可行方向锥中的向量应该保证和等式约束中函数的梯度垂直，这样才能尽量保证 $c_i(x)$, $i \in \mathcal{E}$ 的值不变；而对积极集 $\mathcal{A}(x) \cap \mathcal{I}$ 中的指标 i，沿着该向量 $c_i(x)$ 的值不能增加，因此线性化可行方向对 $c_i(x), i \in \mathcal{A}(x) \cap \mathcal{I}$ 可以是一个下降方向；而对非积极集中的约束，无需提出任何对线性化可行方向的要求.

线性化可行方向锥一般比切锥要大，实际上我们有如下结果：

命题 4.1 设 $c_i(x), i \in \mathcal{E} \cup \mathcal{I}$ 一阶连续可微，则对任意可行点 x 有

$$T_{\mathcal{X}}(x) \subseteq \mathcal{F}(x).$$

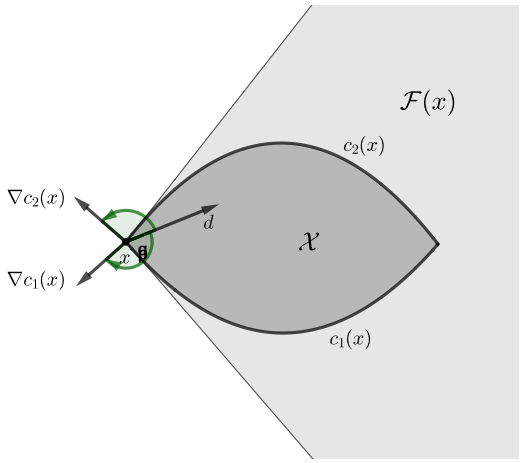


图 4.3 $\mathbb{R}^2$ 上的约束集合和线性化可行方向锥

以上命题的结论反过来是不成立的，我们给出具体的例子. 考虑问题

$$\begin{aligned} \min_{x\in\mathbb{R}} \quad & f(x) = x, \\ \text{s.t.} \quad & c(x) = -x + 3 \leqslant 0. \end{aligned} \tag{4.5.1}$$

根据切锥的定义，可以算出点 $x^* = 3$ 处的切锥为 $T_{\mathcal{X}}(x^*) = \{d \,|\, d \geqslant 0\}$，如图 4.4 所示. 对于线性化可行方向锥，由于 $c'(x^*) = -1$，故 $\mathcal{F}(x^*) = \{d : d \geqslant 0\}$. 此时，我们有 $T_{\mathcal{X}}(x^*) = \mathcal{F}(x^*)$.

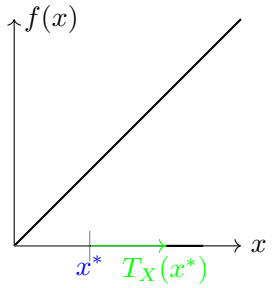


图 4.4 问题 (4.5.1) 的切锥

将问题 (4.5.1) 的约束变形为

$$c(x) = (-x + 3)^3 \leqslant 0.$$

因为可行域没有改变，所以在点 $x^* = 3$ 处，切锥 $T_{\mathcal{X}}(x^*) = \{d : d \geqslant 0\}$ 不变. 对于线性化可行方向锥，由于 $c'(x^*) = -3(x^* - 3)^2 = 0$，所以 $\mathcal{F}(x^*) = \{d \mid d \in \mathbb{R}\}$. 此时，$\mathcal{F}(x^*) \supset T_{\mathcal{X}}(x^*)$（严格包含）. 这个例子告诉我们线性化可行方向锥 $\mathcal{F}(x)$ 不但受到问题可行域 $\mathcal{X}$ 的影响，还会受到 $\mathcal{X}$ 的代数表示方式的影响. 在不改变 $\mathcal{X}$ 的条件下改变定义 $\mathcal{X}$ 的等式（不等式）的数学形式会影响 $\mathcal{F}(x)$ 包含的元素. 而切锥 $T_{\mathcal{X}}(x)$ 的定义直接依赖于可行域 $\mathcal{X}$，因此它不受到 $\mathcal{X}$ 代数表示方式的影响.

线性化可行方向锥容易计算和使用，但会受到问题形式的影响；切锥比较直接地体现了可行域 $\mathcal{X}$ 的性质，但比较难计算. 为了刻画线性化可行方向锥 $\mathcal{F}(x)$ 与切锥 $T_{\mathcal{X}}(x)$ 之间的关系，我们引入约束品性这个概念. 简单来说，大部分的约束品性都是为了保证在最优点 x^* 处，$\mathcal{F}(x^*) = T_{\mathcal{X}}(x^*)$. 这一性质使得我们能够使用 $\mathcal{F}(x)$ 代替 $T_{\mathcal{X}}(x)$，进而更方便地研究约束最优化问题的最优性条件. 这里给出一些常用的约束品性的定义.

定义 4.7 (线性无关约束品性) 给定可行点 x 及相应的积极集 $\mathcal{A}(x)$. 如果积极集对应的约束函数的梯度，即 $\nabla c_i(x), i \in \mathcal{A}(x)$，是线性无关的，则称**线性无关约束品性**（LICQ）在点 x 处成立.

当 LICQ 成立时，切锥和线性化可行方向锥是相同的.

引理 4.2 给定任意可行点 $x \in \mathcal{X}$，如果在该点处 LICQ 成立，则有 $T_{\mathcal{X}}(x) = \mathcal{F}(x)$.

关于 LICQ 的一个常用推广是 Mangasarian–Fromovitz 约束品性，简称为 MFCQ.

定义 4.8 (MFCQ) 给定可行点 x 及相应的积极集 $\mathcal{A}(x)$. 如果存在一个向量 $w \in \mathbb{R}^n$，使得

$$\begin{aligned} \nabla c_i(x)^{\mathrm{T}} w < 0, \quad & \forall i \in \mathcal{A}(x) \cap \mathcal{I}, \\ \nabla c_i(x)^{\mathrm{T}} w = 0, \quad & \forall i \in \mathcal{E}, \end{aligned}$$

并且等式约束对应的梯度集 $\{\nabla c_i(x), i \in \mathcal{E}\}$ 是线性无关的，则称 MFCQ 在点 x 处成立.

可以验证 MFCQ 是 LICQ 的一个弱化版本，即由 LICQ 可以推出 MFCQ，但是反过来不成立. 在 MFCQ 成立的情况下，我们也可以证明 $T_{\mathcal{X}}(x) = \mathcal{F}(x)$，参见 [145][引理 8.2.12].

另外一个用来保证 $T_{\mathcal{X}}(x) = \mathcal{F}(x)$ 的约束品性是线性约束品性.

定义 4.9 (线性约束品性) 如果所有的约束函数 $c_i(x), i \in \mathcal{I} \cup \mathcal{E}$ 都是线性的，则称线性约束品性成立.

当线性约束品性成立时，也有 $T_{\mathcal{X}}(x) = \mathcal{F}(x)$. 因此对只含线性约束的优化问题，例如线性规划、二次规划，很自然地有 $T_{\mathcal{X}}(x) = \mathcal{F}(x), \forall\, x$. 我们无需再关注约束函数的梯度是否线性无关. 一般来说，线性约束品性和 LICQ 之间没有互相包含的关系.

2. Karush-Kuhn-Tucker（KKT）条件

基于几何最优性条件，即定理 4.6，我们想要得到一个计算上更易验证的形式. 切锥和线性化可行方向锥的联系给我们提供了一种方式. 具体地，在定理 4.6 中，如果在局部最优解 x^* 处有

$$T_{\mathcal{X}}(x^*) = \mathcal{F}(x^*)$$

成立（如果 LICQ 在点 x^* 处成立，上式自然满足），那么集合

$$\left\{ d \;\middle|\; \begin{array}{l} d^{\mathrm{T}} \nabla f(x^*) < 0, \\ d^{\mathrm{T}} \nabla c_i(x^*) = 0,\ i \in \mathcal{E}, \\ d^{\mathrm{T}} \nabla c_i(x^*) \leqslant 0,\ i \in \mathcal{A}(x^*) \cap \mathcal{I} \end{array} \right\} \tag{4.5.2}$$

是空集. (4.5.2) 式的验证仍然是非常麻烦的，我们需要将其转化为一个更直接的方式. 这里介绍一个重要的引理，称为 Farkas 引理. 证明见 [98][引理 12.4].

引理 4.3 (Farkas 引理) 设 p 和 q 为两个非负整数，给定向量组 $\{a_i \in \mathbb{R}^n, i = 1,2,\cdots,p\}$, $\{b_i \in \mathbb{R}^n,\ \ i = 1,2,\cdots,q\}$ 和 $c \in \mathbb{R}^n$. 满足以下条件：

$$d^{\mathrm{T}} a_i = 0, \quad i = 1,2,\cdots,p, \tag{4.5.3}$$

$$d^{\mathrm{T}} b_i \geqslant 0, \quad i = 1,2,\cdots,q, \tag{4.5.4}$$

$$d^{\mathrm{T}} c < 0 \tag{4.5.5}$$

的 d 不存在当且仅当存在 $\lambda_i, i = 1,2,\cdots,p$ 和 $\mu_i \geqslant 0, i = 1,2,\cdots,q$，使得

$$c = \sum_{i=1}^{p} \lambda_i a_i + \sum_{i=1}^{q} \mu_i b_i. \tag{4.5.6}$$

利用 Farkas 引理，在 (4.5.3) – (4.5.5) 式中取 $a_i = \nabla c_i(x^*), i \in \mathcal{E}$，$b_i = \nabla c_i(x^*), i \in \mathcal{A}(x^*) \cap \mathcal{I}$ 以及 $c = -\nabla f(x^*)$，集合 (4.5.2) 是空集等价于下式

成立：

$$-\nabla f(x^*) = \sum_{i\in\mathcal{E}} \lambda_i^* \nabla c_i(x^*) + \sum_{i\in\mathcal{A}(x^*)\cap\mathcal{I}} \lambda_i^* \nabla c_i(x^*), \tag{4.5.7}$$

其中 $\lambda_i^* \in \mathbb{R}, i \in \mathcal{E}, \lambda_i^* \geqslant 0, i \in \mathcal{A}(x^*) \cap \mathcal{I}$. 如果补充定义 $\lambda_i^* = 0, i \in \mathcal{I}\backslash\mathcal{A}(x^*)$, 那么

$$-\nabla f(x^*) = \sum_{i\in\mathcal{I}\cup\mathcal{E}} \lambda_i^* \nabla c_i(x^*),$$

这恰好对应于拉格朗日函数关于 x 的一阶最优性条件. 另外，对于任意的 $i \in \mathcal{I}$，我们注意到

$$\lambda_i^* c_i(x^*) = 0.$$

上式称为**互补松弛条件**. 这个条件表明对不等式约束，以下两种情况至少出现一种：乘子 $\lambda_i^* = 0$，或 $c_i(x^*) = 0$（即 $i \in \mathcal{A}(x^*) \cap \mathcal{I}$）. 当以上两种情况恰好只有一种满足时，我们也称此时**严格互补松弛条件** 成立. 一般来说，具有严格互补松弛条件的最优值点有比较好的性质，算法能够很快收敛.

综上所述，我们有如下一阶必要条件，也称作 KKT 条件，并称满足条件(4.5.8)的变量对 (x^*, λ^*) 为 **KKT 对**.

定理 4.7 (KKT 条件 [98]) 假设 x^* 是问题 (4.4.1) 的一个局部最优点. 如果

$$T_{\mathcal{X}}(x^*) = \mathcal{F}(x^*)$$

成立，那么存在拉格朗日乘子 λ_i^* 使得如下条件成立：

$$\begin{aligned}
\text{稳定性条件} \quad & \nabla_x L(x^*, \lambda^*) = \nabla f(x^*) + \sum_{i\in\mathcal{I}\cup\mathcal{E}} \lambda_i^* \nabla c_i(x^*) = 0, \\
\text{原始可行性条件} \quad & c_i(x^*) = 0, \forall i \in \mathcal{E}, \\
\text{原始可行性条件} \quad & c_i(x^*) \leqslant 0, \forall i \in \mathcal{I}, \\
\text{对偶可行性条件} \quad & \lambda_i^* \geqslant 0, \forall i \in \mathcal{I}, \\
\text{互补松弛条件} \quad & \lambda_i^* c_i(x^*) = 0, \forall i \in \mathcal{I}.
\end{aligned} \tag{4.5.8}$$

证明. 因为 $T_{\mathcal{X}}(x^*) = \mathcal{F}(x^*)$，根据定理 4.6，(4.5.2) 式对应的集合为空集. 因此，由 Farkas 引理可知，存在乘子 $\lambda_i^* \in \mathbb{R}, i \in \mathcal{E}, \lambda_i^* \geqslant 0, i \in \mathcal{A}(x^*) \cap \mathcal{I}$，使得 (4.5.7) 式成立. 令 $\lambda_i^* = 0, i \in \mathcal{I}\backslash\mathcal{A}(x^*)$，结合 x^* 的可行性，我们有 (4.5.8) 式成立. □

我们称满足(4.5.8) 式的点 x^* 为 KKT 点. 注意，上面的定理只给出了切锥与线性化可行方向锥相同时的最优性条件. 也就是说，如果在局部最优点

x^* 处 $T_{\mathcal{X}}(x^*) \neq \mathcal{F}(x^*)$，那么 x^* 不一定是 KKT 点. 同样地，因为 KKT 条件只是必要的，所以 KKT 点不一定是局部最优点.

4.5.2 二阶最优性条件

对于问题 (4.4.1)，如果存在一个点 x^* 满足 KKT 条件，我们知道沿着任意线性化可行方向目标函数的一阶近似不会下降. 此时一阶条件无法判断 x^* 是否是最优值点，需要利用二阶信息来进一步判断在其可行邻域内的目标函数值. 具体地，假设 $T_{\mathcal{X}}(x^*) = \mathcal{F}(x^*)$，要判断满足 $d^{\mathrm{T}}\nabla f(x^*) = 0$ 的线性化可行方向 d 是否为 $f(x^*)$ 的下降方向. 我们以拉格朗日函数在这些方向上的曲率信息为桥梁来判断点 x^* 处的最优性. 下面给出临界锥的定义.

定义 4.10 (临界锥) 设 (x^*,λ^*) 满足 KKT 条件(4.5.8)，定义临界锥为

$$\mathcal{C}(x^*,\lambda^*) = \{d \in \mathcal{F}(x^*) \mid \nabla c_i(x^*)^{\mathrm{T}} d = 0, \forall i \in \mathcal{A}(x^*) \cap \mathcal{I} \text{ 且 } \lambda_i^* > 0\},$$

其中 $\mathcal{F}(x^*)$ 为点 x^* 处的线性化可行方向锥.

临界锥是线性化可行方向锥 $\mathcal{F}(x^*)$ 的子集，沿着临界锥中的方向进行优化，所有等式约束和 $\lambda_i^* > 0$ 对应的不等式约束（此时这些不等式约束中的等号均成立）都会尽量保持不变. 注意，对一般的 $d \in \mathcal{F}(x^*)$ 我们仅仅能保证 $d^{\mathrm{T}}\nabla c_i(x^*) \leqslant 0$.

利用上述定义，可得如下结论：

$$d \in \mathcal{C}(x^*,\lambda^*) \Rightarrow \lambda_i^* \nabla c_i(x^*)^{\mathrm{T}} d = 0, \quad \forall i \in \mathcal{E} \cup \mathcal{I}.$$

更进一步地，

$$d \in \mathcal{C}(x^*,\lambda^*) \Rightarrow d^{\mathrm{T}}\nabla f(x^*) = \sum_{i \in \mathcal{E}\cup\mathcal{I}} \lambda_i^* d^{\mathrm{T}} \nabla c_i(x^*) = 0.$$

也就是说，临界锥定义了依据一阶导数不能判断是否为下降或上升方向的线性化可行方向，必须使用高阶导数信息加以判断. 这里给出如下的二阶最优性条件，其具体证明可以在 [98] 中的定理 12.5 以及定理 12.6 中找到：

定理 4.8 (二阶必要条件) 假设 x^* 是问题 (4.4.1) 的一个局部最优解，并且 $T_{\mathcal{X}}(x^*) = \mathcal{F}(x^*)$ 成立. 令 λ^* 为相应的拉格朗日乘子，即 (x^*,λ^*) 满足 KKT 条件，那么

$$d^{\mathrm{T}}\nabla_{xx}^2 L(x^*,\lambda^*) d \geqslant 0, \quad \forall d \in \mathcal{C}(x^*,\lambda^*).$$

定理 4.9 (二阶充分条件) 假设在可行点 x^* 处，存在一个拉格朗日乘子 λ^*，使得 (x^*,λ^*) 满足 KKT 条件. 如果

$$d^{\mathrm{T}}\nabla^2_{xx}L(x^*,\lambda^*)d>0,\quad \forall d\in\mathcal{C}(x^*,\lambda^*),\ d\neq 0,$$

那么 x^* 为问题 (4.4.1) 的一个严格局部极小解.

我们比对无约束优化问题的二阶最优性条件（定理4.3）不难发现，约束优化问题的二阶最优性条件也要求某种“正定性”，但只需要考虑临界锥 $\mathcal{C}(x^*,\lambda^*)$ 中的向量而无需考虑全空间的向量. 因此有些教材中又将其称为“投影半正定性”.

为了更深刻地理解约束优化的最优性理论，我们考虑一个具体的例子. 给定如下约束优化问题：

$$\min\quad x_1^2+x_2^2,\quad \text{s.t.}\quad \frac{x_1^2}{4}+x_2^2-1=0,$$

其拉格朗日函数为

$$L(x,\lambda)=x_1^2+x_2^2+\lambda(\frac{x_1^2}{4}+x_2^2-1).$$

该问题可行域在任意一点 $x=(x_1,x_2)^{\mathrm{T}}$ 处的线性化可行方向锥为

$$\mathcal{F}(x)=\{(d_1,d_2)\mid \frac{x_1}{4}d_1+x_2d_2=0\}.$$

因为只有一个等式约束且其对应函数的梯度非零，故有 LICQ 成立，且在 KKT 对 (x,λ) 处有 $\mathcal{C}(x,\lambda)=\mathcal{F}(x)$. 可以计算出其 4 个 KKT 对

$$(x^{\mathrm{T}},\lambda)=(2,0,-4),\quad (-2,0,-4),\quad (0,1,-1)\quad 和\quad (0,-1,-1).$$

我们考虑第一个 KKT 对 $y=(2,0,-4)^{\mathrm{T}}$ 和第三个 KKT 对 $z=(0,1,-1)^{\mathrm{T}}$. 计算可得，

$$\nabla^2_{xx}L(y)=\begin{bmatrix}0&0\\0&-6\end{bmatrix},\quad \mathcal{C}(y)=\{(d_1,d_2)\mid d_1=0\}.$$

取 $d=(0,1)$，则

$$d^{\mathrm{T}}\nabla^2_{xx}L(y)d=-6<0,$$

因此 y 不是局部最优点. 类似地，对于 KKT 对 $z=(0,1,-1)$，

$$\nabla^2_{xx}L(z)=\begin{bmatrix}\frac{3}{2}&0\\0&0\end{bmatrix},\quad \mathcal{C}(z)=\{(d_1,d_2)\mid d_2=0\}.$$

对于任意的 $d=(d_1,0)$ 且 $d_1\neq 0$，

$$d^{\mathrm{T}}\nabla^2_{xx}L(z)d=\frac{3}{2}d_1^2>0.$$

因此，z 为一个严格局部最优点.

4.6 带约束凸优化问题的最优性理论

在实际问题中，优化问题 (4.4.1) 的目标函数和约束函数往往是凸的（可能不可微）. 因此，凸优化问题的最优性条件的研究具有重要意义. 这里考虑如下形式的凸优化问题：

$$\begin{aligned}\min_{x\in\mathcal{D}}\quad & f(x),\\ \text{s.t.}\quad & c_i(x)\leqslant 0,\quad i=1,2,\cdots,m,\\ & Ax=b,\end{aligned}\tag{4.6.1}$$

其中 $f(x)$ 为适当的凸函数，$c_i(x), i=1,2,\cdots,m$ 是凸函数且 $\mathbf{dom}\, c_i=\mathbb{R}^n$，以及 $A\in\mathbb{R}^{p\times n}, b\in\mathbb{R}^p$ 是已知的. 我们用集合 $\mathcal{D}$ 表示自变量 x 的自然定义域，即

$$\mathcal{D}=\mathbf{dom}\, f=\{x\mid f(x)<+\infty\}.$$

自变量 x 除了受到自然定义域的限制以外，还需要受到约束的限制. 我们定义可行域

$$\mathcal{X}=\{x\in\mathcal{D}\,:\,c_i(x)\leqslant 0, i=1,2,\cdots,m; Ax=b\}.$$

在这里注意，由于凸优化问题的可行域是凸集，因此等式约束只可能是线性约束. 凸优化问题 (4.6.1) 有很多好的性质. 一个自然的问题是：我们能否像研究无约束问题那样找到该问题最优解的一阶充要条件？如果这样的条件存在，它在什么样的约束品性下成立？本节将比较具体地回答这一问题.

4.6.1 Slater 约束品性与强对偶原理

在通常情况下，优化问题的对偶间隙都是大于 0 的，即强对偶原理不满足. 但是，对于很多凸优化问题，在特定约束品性满足的情况下可以证明强对偶原理. 简单直观的一种约束品性是存在满足所有约束条件的严格可行解. 首先，我们给出集合 $\mathcal{D}$ 的相对内点集 $\mathbf{relint}\,\mathcal{D}$ 的定义.

定义 4.11 (相对内点集) 给定集合 $\mathcal{D}$，记其仿射包为 $\mathbf{affine}\,\mathcal{D}$（见定义2.14）. 集合 $\mathcal{D}$ 的**相对内点集** 定义为

$$\mathbf{relint}\,\mathcal{D} = \{x \in \mathcal{D} \mid \exists\, r > 0, \text{使得 } B(x,r) \cap \mathbf{affine}\,\mathcal{D} \subseteq \mathcal{D}\}.$$

相对内点是内点的推广，我们知道若 x 是集合 $\mathcal{D} \subseteq \mathbb{R}^n$ 的内点，则存在一个以 x 为球心的 n 维球含于集合 $\mathcal{D}$. 若 $\mathcal{D}$ 本身的 "维数" 较低，则 $\mathcal{D}$ 不可能有内点；但如果在它的仿射包 $\mathbf{affine}\,\mathcal{D}$ 中考虑，则 $\mathcal{D}$ 可能有相对内点.

借助相对内点的定义，我们给出 Slater 约束品性.

定义 4.12 (Slater 约束品性) 若对凸优化问题 (4.6.1)，存在 $x \in \mathbf{relint}\,\mathcal{D}$ 满足

$$c_i(x) < 0, \quad i = 1,2,\cdots,m, \quad Ax = b,$$

则称对此问题 Slater 约束品性满足. 有时也称该约束品性为Slater **条件**.

Slater 约束品性实际上是要求自然定义域 $\mathcal{D}$ 的相对内点中存在使得不等式约束严格成立的点. 对于很多凸优化问题，自然定义域 $\mathcal{D}$ 的仿射包 $\mathbf{affine}\,\mathcal{D} = \mathbb{R}^n$，在这种情况下 Slater 条件中的相对内点就是内点.

注 4.1 当一些不等式约束是仿射函数时，Slater 条件可以适当放宽. 不妨假设前 k 个不等式约束是仿射函数，此时 Slater 约束品性可变为：存在 $x \in \mathbf{relint}\,\mathcal{D}$ 满足

$$c_i(x) \leqslant 0, \quad i = 1,2,\cdots,k; \quad c_i(x) < 0, \quad i = k+1,k+2\cdots,m; \quad Ax = b,$$

即对线性不等式约束无需要求其存在严格可行点.

若凸优化问题 (4.6.1) 满足 Slater 条件，一个很重要的结论就是强对偶原理成立. 此外当对偶问题最优值 $d^* > -\infty$ 时，对偶问题的最优解可以取到，即存在对偶可行解 (λ^*, ν^*)，满足 $g(\lambda^*, \nu^*) = d^* = p^*$. 实际上我们有下面的定理：

定理 4.10 (强对偶原理 [20]第 5.3.2 小节) 如果凸优化问题(4.6.1)满足 Slater 条件，则强对偶原理成立.

4.6.2 一阶充要条件

对于一般的约束优化问题，当问题满足特定约束品性时，我们知道 KKT 条件是局部最优解处的必要条件. 而对于凸优化问题，当 Slater 条件满足时，

KKT 条件则变为局部最优解的充要条件（根据凸性，局部最优解也是全局最优解）. 实际上我们有如下定理.

定理 4.11 (凸问题 KKT 条件) 对于凸优化问题 (4.6.1)，如果 Slater 条件成立，那么 x^*,λ^* 分别是原始、对偶全局最优解当且仅当

$$\begin{aligned}
\text{稳定性条件} \quad & 0 \in \partial f(x^*) + \sum_{i\in\mathcal{I}} \lambda_i^* \partial c_i(x^*) + \sum_{i\in\mathcal{E}} \lambda_i^* a_i, \\
\text{原始可行性条件} \quad & Ax^* = b, \forall i \in \mathcal{E}, \\
\text{原始可行性条件} \quad & c_i(x^*) \leqslant 0, \forall i \in \mathcal{I}, \\
\text{对偶可行性条件} \quad & \lambda_i^* \geqslant 0, \forall i \in \mathcal{I}, \\
\text{互补松弛条件} \quad & \lambda_i^* c_i(x^*) = 0, \forall i \in \mathcal{I}.
\end{aligned} \tag{4.6.2}$$

其中 a_i 是矩阵 A^{T} 的第 i 列.

在这里条件 (4.6.2) 和条件(4.5.8) 略有不同. 在凸优化问题中没有假设 $f(x)$ 和 $c_i(x)$ 是可微函数，因此我们在这里使用的是次梯度. 当 $f(x)$ 和 $c_i(x)$ 都是凸可微函数时，条件 (4.6.2) 就是条件 (4.5.8).

定理 4.11 的充分性比较容易说明. 实际上，设存在 $(\bar{x},\bar{\lambda})$ 满足 KKT 条件 (4.6.2)，我们考虑凸优化问题的拉格朗日函数

$$L(x,\lambda) = f(x) + \sum_{i\in\mathcal{I}} \lambda_i c_i(x) + \sum_{i\in\mathcal{E}} \lambda_i (a_i^{\mathrm{T}} x - b_i),$$

当固定 $\lambda = \bar{\lambda}$ 时，注意到 $\bar{\lambda}_i \geqslant 0, i \in \mathcal{I}$ 以及 $\bar{\lambda}_i (a_i^{\mathrm{T}} x), i \in \mathcal{E}$ 是线性函数可知 $L(x,\bar{\lambda})$ 是关于 x 的凸函数. 由凸函数全局最优点的一阶充要性可知，此时 $\bar{x}$ 就是 $L(x,\bar{\lambda})$ 的全局极小点. 根据拉格朗日对偶函数的定义，

$$L(\bar{x},\bar{\lambda}) = \inf_{x\in\mathcal{D}} L(x,\bar{\lambda}) = g(\bar{\lambda}).$$

根据原始可行性条件 $A\bar{x} = b$ 以及互补松弛条件 $\bar{\lambda}_i c_i(\bar{x}) = 0, i \in \mathcal{I}$ 可以得到

$$L(\bar{x},\bar{\lambda}) = f(\bar{x}) + 0 + 0 = f(\bar{x}).$$

根据弱对偶原理，

$$L(\bar{x},\bar{\lambda}) = f(\bar{x}) \geqslant p^* \geqslant d^* \geqslant g(\bar{\lambda}). \tag{4.6.3}$$

由于 $L(\bar{x},\bar{\lambda}) = g(\bar{\lambda})$，(4.6.3) 式中的不等号皆为等号，因此我们有 $p^* = d^*$ 且 $\bar{x},\bar{\lambda}$ 分别是原始问题和对偶问题的最优解.

定理 4.11 的充分性说明，若能直接求解出凸优化问题 (4.6.1) 的 KKT 对，则其就对应 (4.6.1) 的最优解. 注意，在充分性部分的证明中，我们没有使用 Slater 条件，这是因为在证明的一开始假设了 KKT 点是存在的. Slater 条件的意义在于当问题 (4.6.1) 最优解存在时，其相应 KKT 条件也会得到满足. 换句话说，当 Slater 条件不满足时，即使原始问题存在全局极小值点，也可能不存在 (x^*,λ^*) 满足 KKT 条件(4.6.2).

定理4.11的必要性证明比较复杂，我们在此书中不作具体讨论.

4.7 约束优化最优性理论应用实例

这一部分，我们以实例的方式更进一步地解释光滑凸优化问题、非光滑凸优化问题以及光滑非凸优化问题的最优性条件.

4.7.1 仿射空间的投影问题

考虑优化问题

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \frac{1}{2}\|x-y\|_2^2, \\ \text{s.t.} \quad & Ax=b, \end{aligned}$$

其中 $A\in\mathbb{R}^{m\times n}$, $b\in\mathbb{R}^m$ 以及 $y\in\mathbb{R}^n$ 为给定的矩阵和向量. 这里不妨设矩阵 A 是行满秩的（否则，可以按照 A 的行之间的相关性消除约束冗余，从而使得消去后的 $\tilde{A}$ 是行满秩的）. 这个问题可以看成仿射平面 $\{x\in\mathbb{R}^n \mid Ax=b\}$ 的投影问题.

对于等式约束，我们引入拉格朗日乘子 $\lambda\in\mathbb{R}^m$，构造拉格朗日函数

$$L(x,\lambda)=\frac{1}{2}\|x-y\|^2+\lambda^{\mathrm{T}}(Ax-b).$$

因为只有仿射约束，故 Slater 条件满足. x^* 为一个全局最优解，当且仅当存在 $\lambda^*\in\mathbb{R}^m$ 使得

$$\begin{cases} x^*-y+A^{\mathrm{T}}\lambda=0, \\ Ax^*=b. \end{cases}$$

由上述 KKT 条件第一式，等号左右两边同时左乘 A 可得

$$Ax^*-Ay+AA^{\mathrm{T}}\lambda=0.$$

注意到 $Ax^* = b$ 以及 AA^{T} 是可逆矩阵，因此可解出乘子

$$\lambda = (AA^{\mathrm{T}})^{-1}(Ay - b),$$

将 λ 代回 KKT 条件第一式可知

$$x^* = y - A^{\mathrm{T}}(AA^{\mathrm{T}})^{-1}(Ay - b).$$

因此，点 y 到集合 $\{x \mid Ax = b\}$ 的投影为 $y - A^{\mathrm{T}}(AA^{\mathrm{T}})^{-1}(Ay - b)$.

4.7.2 线性规划问题

考虑线性规划问题

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^{\mathrm{T}}x, \\ \text{s.t.} \quad & Ax = b, \\ & x \geqslant 0, \end{aligned} \tag{4.7.1}$$

其中 $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n$ 分别为给定的矩阵和向量.

线性规划问题(4.7.1) 的拉格朗日函数为

$$\begin{aligned} L(x,s,\nu) &= c^{\mathrm{T}}x + \nu^{\mathrm{T}}(Ax - b) - s^{\mathrm{T}}x \\ &= -b^{\mathrm{T}}\nu + (A^{\mathrm{T}}\nu - s + c)^{\mathrm{T}}x, \quad s \geqslant 0, \end{aligned}$$

其中 $s \in \mathbb{R}^n, \nu \in \mathbb{R}^m$. 由于线性规划是凸问题且满足 Slater 条件，因此对于任意一个全局最优解 x^*，我们有如下 KKT 条件：

$$\begin{cases} c + A^{\mathrm{T}}\nu^* - s^* = 0, \\ Ax^* = b, \\ x^* \geqslant 0, \\ s^* \geqslant 0, \\ s^* \odot x^* = 0, \end{cases} \tag{4.7.2}$$

其中 $\odot$ 表示向量的 Hadamard 积，即 $(x \odot y)_i = x_i y_i$. 上述 KKT 条件也是充分的，即满足上式的 x^* 也为问题(4.7.1)的全局最优解，并且 s^*, ν^* 也为其对偶问题的全局最优解.

我们可以进一步说明线性规划问题的解和其对偶问题的解之间的关系. 设原始问题和对偶问题最优解处函数值分别为 p^* 和 d^*，则根据 p^* 取值情况有如下三种可能：

(1) 如果 $-\infty < p^* < +\infty$（有界），那么原始问题可行而且存在最优解. 由 Slater 条件知强对偶原理成立，因此有 $d^* = p^*$，即对偶问题也是可行的且存在最优解.

(2) 如果 $p^* = -\infty$，那么原始问题可行，但目标函数值无下界. 由弱对偶原理知 $d^* \leqslant p^* = -\infty$，即 $d^* = -\infty$. 因为对偶问题是对目标函数极大化，所以此时对偶问题不可行.

(3) 如果 $p^* = +\infty$, 那么原始问题无可行解. 注意到 Slater 条件对原始问题不成立，此时对偶问题既可能函数值无界（对应 $d^* = +\infty$），也可能无可行解（对应 $d^* = -\infty$）. 我们指出，此时不可能出现 $-\infty < d^* < +\infty$ 的情形，这是因为如果对偶问题可行且存在最优解，那么可对对偶问题应用强对偶原理，进而导出原始问题也存在最优解，这与 $p^* = +\infty$ 矛盾.

最后，我们将原始问题和对偶问题解的关系总结在表 4.1 中. 可以看到，针对线性规划问题及其对偶问题，解的情况只有四种可能的组合.

表 4.1 线性规划原始问题和对偶问题的对应关系

原始问题	对偶问题		
	有界	无界	不可行
有界	✓	×	×
无界	×	×	✓
不可行	×	✓	✓

4.8 总结

本章介绍了约束和无约束优化问题的最优性理论，并对于凸优化问题的理论给出了进一步的分析. 关于一般优化问题的理论，更多内容以及本章省略的证明和细节可以在 [98,120] 中找到.

对于非光滑非凸问题的最优性条件，我们一般借助其 Fréchet 次微分来研究. 当函数光滑的时候，该次微分就是正常的梯度. 对非光滑凸函数，Fréchet 次微分就是其次梯度的集合. 对于非光滑非凸的情形，我们则用

Fréchet 次微分来判断最优性条件（考虑无约束的情形，任一局部最优解 x^* 处的 Fréchet 次微分必定包含零向量）. 相关内容读者可以参考 [91].

优化问题的对偶问题与其目标函数的共轭函数的极值问题在某种程度上是等价的. 对于更多形式的广义不等式约束优化问题及其拉格朗日函数对偶推导，以及更详细的凸优化问题的理论，读者可以参考 [20,112].

本章凸优化问题的最优性理论部分的编写参考了 [20,113]，一般光滑问题的最优性理论的编写参考了 [98]，对偶理论的编写参考了 Lieven Vandenberghe 教授的课件，更多细节和解释可以在其中找到.

最后，我们在表 4.2 和表 4.3 中总结本章涉及的无约束优化问题和约束优化问题的最优性条件和所需的约束品性（仅约束优化问题），其中 "---" 表示无需考虑（凸问题）或本书未进行讨论（非凸问题）的条件. 在实际应用中读者需要根据问题种类使用合适的最优性条件对问题进行理论分析和算法构建.

表 4.2 无约束优化问题及其最优性条件

问题	一阶条件	二阶条件
可微问题	$\nabla f(x^*)=0$（必要）	$\nabla^2 f(x^*)\succeq 0$（必要） $\nabla^2 f(x^*)\succ 0$（充分）
凸问题	$0\in\partial f(x^*)$（充要）	—
复合优化问题	$-\nabla f(x^*)\in\partial h(x^*)$（必要）	—

表 4.3 约束优化问题的最优性条件和相应约束品性

问题	一阶条件	二阶条件	约束品性
一般问题	KKT 条件（必要）	定理 4.8（必要） 定理 4.9（充分）[1]	LICQ[2]
凸问题	KKT 条件（充要）	—	Slater

习题 5

4.1 考虑优化问题

$$\min_{x\in\mathbb{R}^n} \quad x^{\mathrm{T}}Ax+2b^{\mathrm{T}}x,$$

其中 $A \in \mathcal{S}^n$, $b \in \mathbb{R}^n$. 为了保证该问题最优解存在，A,b 需要满足什么性质?

4.2 试举例说明对无约束光滑优化问题，二阶必要条件不是充分的，二阶充分条件也不是必要的（见定理4.3）.

4.3 证明下列锥是自对偶锥：

(a) 半正定锥 $\{X \mid X \succeq 0\}$（全空间为 $\mathcal{S}^n$）；

(b) 二次锥 $\{(x,t) \in \mathbb{R}^{n+1} \mid t \geqslant \|x\|_2\}$（全空间为 $\mathbb{R}^{n+1}$）.

4.4 考虑优化问题

$$\min_{x \in \mathbb{R}^n} \quad x^{\mathrm{T}} A x + 2b^{\mathrm{T}} x, \quad \text{s.t.} \quad \|x\|_2 \leqslant \Delta,$$

其中 $A \in \mathcal{S}^n_{++}$, $b \in \mathbb{R}^n$, $\Delta > 0$. 求出该问题的最优解.

4.5 考虑函数 $f(x) = 2x_1^2 + x_2^2 - 2x_1x_2 + 2x_1^3 + x_1^4$，求出其所有一阶稳定点，并判断它们是否为局部最优点（极小或极大）、鞍点或全局最优点?

4.6 给出下列优化问题的显式解：

(a) $\min\limits_{x \in \mathbb{R}^n} \quad c^{\mathrm{T}} x, \quad \text{s.t.} \quad Ax = b$，其中 $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$;

(b) $\min\limits_{x \in \mathbb{R}^n} \quad \|x\|_2, \quad \text{s.t.} \quad Ax = b$;

(c) $\min\limits_{x \in \mathbb{R}^n} \quad c^{\mathrm{T}} x, \quad \text{s.t.} \quad \mathbf{1}^{\mathrm{T}} x = 1,\ x \geqslant 0$;

(d) $\min\limits_{X \in \mathbb{R}^{m \times n}} \quad \|X\|_* + \dfrac{1}{2}\|X - Y\|_F^2$，其中 $Y \in \mathbb{R}^{m \times n}$ 是已知的.

4.7 计算下列优化问题的对偶问题.

(a) $\min\limits_{x \in \mathbb{R}^n} \quad \|x\|_1, \quad \text{s.t.} \quad Ax = b$;

(b) $\min\limits_{x \in \mathbb{R}^n} \quad \|Ax - b\|_1$;

(c) $\min\limits_{x \in \mathbb{R}^n} \quad \|Ax - b\|_\infty$;

(d) $\min\limits_{x \in \mathbb{R}^n} \quad x^{\mathrm{T}} A x + 2b^{\mathrm{T}} x, \quad \text{s.t.} \quad \|x\|_2^2 \leqslant 1$, 其中 A 为正定矩阵.

---

[1] 一般约束优化问题的二阶充分条件不需要 LICQ 作为前提.

[2] 或其他可推出 $T_{\mathcal{X}}(x^*) = \mathcal{F}(x^*)$ 的约束品性.

4.8 如下论断正确吗？为什么？
对等式约束优化问题

$$\begin{aligned} \min \quad & f(x), \\ \text{s.t.} \quad & c_i(x) = 0,\ i \in \mathcal{E}. \end{aligned}$$

考虑与之等价的约束优化问题：

$$\begin{aligned} \min \quad & f(x), \\ \text{s.t.} \quad & c_i^2(x) = 0,\ i \in \mathcal{E}. \end{aligned} \tag{4.8.1}$$

设 $x^\sharp$ 是上述问题的一个 KKT 点，根据(4.5.8)式，$x^\sharp$ 满足

$$\begin{aligned} 0 &= \nabla f(x^\sharp) + 2\sum_{i\in\mathcal{E}} \lambda_i^\sharp c_i(x^\sharp)\nabla c_i(x^\sharp), \\ 0 &= c_i(x^\sharp), \quad i \in \mathcal{E}, \end{aligned}$$

其中 $\lambda_i^\sharp$ 是相应的拉格朗日乘子. 整理上式得 $\nabla f(x^\sharp) = 0$. 这说明对等式约束优化问题，我们依然能给出类似无约束优化问题的最优性条件.

4.9 证明：若在点 x 处线性约束品性（见定义 4.9）满足，则有 $T_{\mathcal{X}}(x) = \mathcal{F}(x)$.

4.10 考虑优化问题

$$\begin{aligned} \min_{x\in\mathbb{R}^2} \quad & x_1, \\ \text{s.t.} \quad & 16 - (x_1 - 4)^2 - x_2^2 \geqslant 0, \\ & x_1^2 + (x_2 - 2)^2 - 4 = 0. \end{aligned}$$

求出该优化问题的 KKT 点，并判断它们是否是局部极小点、鞍点以及全局极小点？

4.11 考虑等式约束的最小二乘问题

$$\min_{x\in\mathbb{R}^n} \quad \|Ax - b\|_2^2, \quad \text{s.t.} \quad Gx = h,$$

其中 $A \in \mathbb{R}^{m\times n}$ 且 $\operatorname{rank}(A) = n$，$G \in \mathbb{R}^{p\times n}$ 且 $\operatorname{rank}(G) = p$.

(a) 写出该问题的对偶问题；

(b) 给出原始问题和对偶问题的最优解的显式表达式.

4.12 考虑优化问题

$$\min_{x\in\mathbb{R}^n} \quad x^{\mathrm{T}}Ax + 2b^{\mathrm{T}}x, \quad \text{s.t.} \quad \|x\|_2 \leqslant 1,$$

其中 $A \in \mathcal{S}^n$, $b \in \mathbb{R}^n$. 写出该问题的对偶问题，以及对偶问题的对偶问题.

4.13 考虑支持向量机问题

$$\begin{aligned} \min_{x\in\mathbb{R}^n,\xi} \quad & \frac{1}{2}\|x\|_2^2 + \mu\sum_{i=1}^{m}\xi_i, \\ \text{s.t.} \quad & b_i a_i^{\mathrm{T}} x \geqslant 1 - \xi_i,\ i = 1,2,\cdots,m, \\ & \xi_i \geqslant 0,\ i = 1,2,\cdots,m, \end{aligned}$$

其中 $\mu > 0$ 为常数且 $b_i \in \mathbb{R}$, $a_i \in \mathbb{R}^n$, $i = 1,2,\cdots,m$ 是已知的. 写出该问题的对偶问题.

4.14 考虑优化问题

$$\min_{x\in\mathbb{R},y>0} \quad e^{-x}, \quad \text{s.t.} \quad \frac{x^2}{y} \leqslant 0.$$

(a) 证明这是一个凸优化问题，求出最小值并判断 Slater 条件是否成立；

(b) 写出该问题的对偶问题，并求出对偶问题的最优解以及对偶间隙.

第五章　无约束优化算法

本章考虑如下无约束优化问题：

$$\min_{x\in\mathbb{R}^n} \quad f(x), \tag{5.0.1}$$

其中 $f(x)$ 是 $\mathbb{R}^n \to \mathbb{R}$ 的函数．无约束优化问题是众多优化问题中最基本的问题，它对自变量 x 的取值范围不加限制，所以无需考虑 x 的可行性．对于光滑函数，我们可以较容易地利用梯度和海瑟矩阵的信息来设计算法；对于非光滑函数，我们可以利用次梯度来构造迭代格式．很多无约束优化问题的算法思想可以推广到其他优化问题上，因此掌握如何求解无约束优化问题的方法是设计其他优化算法的基础．

无约束优化问题的优化算法主要分为两大类：线搜索类型的优化算法和信赖域类型的优化算法．它们都是对函数 $f(x)$ 在局部进行近似，但处理近似问题的方式不同．线搜索类算法根据搜索方向的不同可以分为梯度类算法、次梯度算法、牛顿算法、拟牛顿算法等．一旦确定了搜索的方向，下一步即沿着该方向寻找下一个迭代点．而信赖域算法主要针对 $f(x)$ 二阶可微的情形，它是在一个给定的区域内使用二阶模型近似原问题，通过不断直接求解该二阶模型从而找到最优值点．我们在本章中将初步介绍这些算法的思想和具体执行过程，并简要分析它们的性质．

5.1　线搜索方法

对于优化问题(5.0.1)，我们将求解 $f(x)$ 的最小值点的过程比喻成下山的过程．假设一个人处于某点 x 处，$f(x)$ 表示此地的高度，为了寻找最低点，在点 x 处需要确定如下两件事情：第一，下一步该向哪一方向行走；第二，沿着该方向行走多远后停下以便选取下一个下山方向．以上这两个因素确定后，便可以一直重复，直至到达 $f(x)$ 的最小值点．

线搜索类算法的数学表述为：给定当前迭代点 x^k，首先通过某种算法选取向量 d^k，之后确定正数 α_k，则下一步的迭代点可写作

$$x^{k+1} = x^k + \alpha_k d^k. \tag{5.1.1}$$

我们称 d^k 为迭代点 x^k 处的**搜索方向**，α_k 为相应的**步长**. 这里要求 d^k 是一个**下降方向**，即 $(d^k)^{\mathrm{T}}\nabla f(x^k) < 0$. 这个下降性质保证了沿着此方向搜索函数 f 的值会减小. 线搜索类算法的关键是如何选取一个好的方向 $d^k \in \mathbb{R}^n$ 以及合适的步长 α_k.

在本节中，我们将回答如何选取 α_k 这一问题. 这是因为选取 d^k 的方法千差万别，但选取 α_k 的方法在不同算法中非常相似. 首先构造辅助函数

$$\phi(\alpha) = f(x^k + \alpha d^k),$$

其中 d^k 是给定的下降方向，$\alpha > 0$ 是该辅助函数的自变量. 函数 $\phi(\alpha)$ 的几何含义非常直观：它是目标函数 $f(x)$ 在射线 $\{x^k + \alpha d^k \,:\, \alpha > 0\}$ 上的限制. 注意到 $\phi(\alpha)$ 是一个一元函数，而我们研究一元函数相对比较方便.

线搜索的目标是选取合适的 α_k 使得 $\phi(\alpha_k)$ 尽可能减小. 但这一工作并不容易：α_k 应该使得 f 充分下降，与此同时不应在寻找 α_k 上花费过多的计算量. 我们需要权衡这两个方面. 一个自然的想法是寻找 α_k 使得

$$\alpha_k = \arg\min_{\alpha>0} \phi(\alpha),$$

即 α_k 为最佳步长. 这种线搜索算法被称为**精确线搜索算法**. 需要指出的是，使用精确线搜索算法时我们可以在多数情况下得到优化问题的解，但选取 α_k 通常需要很大计算量，在实际应用中较少使用. 另一个想法不要求 α_k 是 $\phi(\alpha)$ 的最小值点，而是仅仅要求 $\phi(\alpha_k)$ 满足某些不等式性质. 这种线搜索方法被称为**非精确线搜索算法**. 由于非精确线搜索算法结构简单，在实际应用中较为常见，接下来我们介绍该算法的结构.

5.1.1 线搜索准则

在非精确线搜索算法中，选取 α_k 需要满足一定的要求，这些要求被称为**线搜索准则**. 这里指出，线搜索准则的合适与否直接决定了算法的收敛性，若选取不合适的线搜索准则将会导致算法无法收敛. 为此我们给出一个例子.

例 5.1 考虑一维无约束优化问题

$$\min_x \quad f(x)=x^2,$$

迭代初始点 $x^0=1$. 由于问题是一维的，下降方向只有 $\{-1,+1\}$ 两种. 我们选取 $d^k=-\text{sign}(x^k)$，且只要求选取的步长满足迭代点处函数值单调下降，即 $f(x^k+\alpha_k d^k)<f(x^k)$. 考虑选取如下两种步长：

$$\alpha_{k,1}=\frac{1}{3^{k+1}},\quad \alpha_{k,2}=1+\frac{2}{3^{k+1}},$$

通过简单计算可以得到

$$x_1^k=\frac{1}{2}\left(1+\frac{1}{3^k}\right),\quad x_2^k=\frac{(-1)^k}{2}\left(1+\frac{1}{3^k}\right).$$

显然，序列 $\{f(x_1^k)\}$ 和序列 $\{f(x_2^k)\}$ 均单调下降，但序列 $\{x_1^k\}$ 收敛的点不是极小值点，序列 $\{x_2^k\}$ 则在原点左右振荡，不存在极限.

出现上述情况的原因是在迭代过程中函数值 $f(x^k)$ 的下降量不够充分，以至于算法无法收敛到极小值点. 为了避免这种情况发生，必须引入一些更合理的线搜索准则来确保迭代的收敛性.

1. Armijo 准则

我们首先引入 Armijo 准则，它是一个常用的线搜索准则. 引入 Armijo 准则的目的是保证每一步迭代充分下降.

定义 5.1 (Armijo 准则) 设 d^k 是点 x^k 处的下降方向，若

$$f(x^k+\alpha d^k)\leqslant f(x^k)+c_1\alpha\nabla f(x^k)^{\mathrm{T}}d^k, \tag{5.1.2}$$

则称步长 α 满足 **Armijo 准则**，其中 $c_1\in(0,1)$ 是一个常数.

Armijo 准则(5.1.2)有非常直观的几何含义，它指的是点 $(\alpha,\phi(\alpha))$ 必须在直线

$$l(\alpha)=\phi(0)+c_1\alpha\nabla f(x^k)^{\mathrm{T}}d^k$$

的下方. 如图 5.1 所示，区间 $[0,\alpha_1]$ 中的点均满足 Armijo 准则. 我们注意到 d^k 为下降方向，这说明 $l(\alpha)$ 的斜率为负，选取符合条件(5.1.2)的 α 确实会使得函数值下降. 在实际应用中，参数 c_1 通常选为一个很小的正数，例如

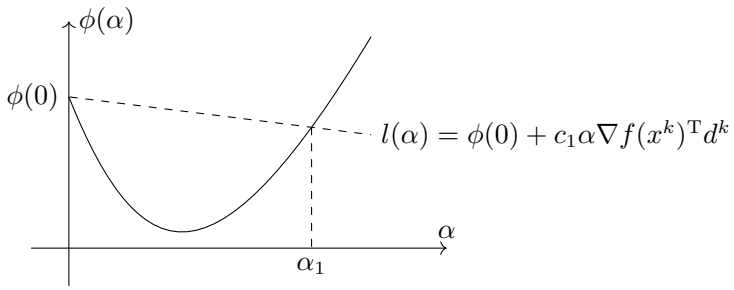


图 5.1 Armijo 准则

$c_1 = 10^{-3}$，这使得 Armijo 准则非常容易得到满足．但是仅仅使用 Armijo 准则并不能保证迭代的收敛性，这是因为 $\alpha = 0$ 显然满足条件(5.1.2)，而这意味着迭代序列中的点固定不变，研究这样的步长是没有意义的．为此，Armijo 准则需要配合其他准则共同使用．

在优化算法的实现中，寻找一个满足 Armijo 准则的步长是比较容易的，一个最常用的算法是**回退法**．给定初值 $\hat{\alpha}$，回退法通过不断以指数方式缩小试探步长，找到第一个满足 Armijo 准则(5.1.2)的点．具体来说，回退法选取

$$\alpha_k = \gamma^{j_0}\hat{\alpha},$$

其中

$$j_0 = \min\{j = 0,1,\cdots \mid f(x^k + \gamma^j\hat{\alpha}d^k) \leqslant f(x^k) + c_1\gamma^j\hat{\alpha}\nabla f(x^k)^{\mathrm{T}}d^k\},$$

参数 $\gamma \in (0,1)$ 为一个给定的实数．回退法的基本过程如算法5.1所示．

算法 5.1 线搜索回退法

1. 选择初始步长 $\hat{\alpha}$，参数 $\gamma, c \in (0,1)$．初始化 $\alpha \leftarrow \hat{\alpha}$．
2. **while** $f(x^k + \alpha d^k) > f(x^k) + c\alpha\nabla f(x^k)^{\mathrm{T}}d^k$ **do**
3. 令 $\alpha \leftarrow \gamma\alpha$．
4. **end while**
5. 输出 $\alpha_k = \alpha$．

该算法被称为回退法是因为 α 的试验值是由大至小的，它可以确保输出的 α_k 能尽量地大．此外算法5.1不会无限进行下去，因为 d^k 是一个下降方

向，当 α 充分小时，Armijo 准则总是成立的．在实际应用中我们通常也会给 α 设置一个下界，防止步长过小．

2. Goldstein 准则

为了克服 Armijo 准则的缺陷，我们需要引入其他准则来保证每一步的 α^k 不会太小．既然 Armijo 准则只要求点 $(\alpha,\phi(\alpha))$ 必须处在某直线下方，我们也可使用相同的形式使得该点必须处在另一条直线的上方．这就是 Armijo-Goldstein 准则，简称 Goldstein 准则．

定义 5.2 (Goldstein 准则) 设 d^k 是点 x^k 处的下降方向，若

$$f(x^k+\alpha d^k) \leqslant f(x^k)+c\alpha\nabla f(x^k)^{\mathrm{T}}d^k, \tag{5.1.3a}$$

$$f(x^k+\alpha d^k) \geqslant f(x^k)+(1-c)\alpha\nabla f(x^k)^{\mathrm{T}}d^k, \tag{5.1.3b}$$

则称步长 α 满足 **Goldstein 准则**，其中 $c\in\left(0,\frac{1}{2}\right)$.

同样，Goldstein 准则 (5.1.3) 也有非常直观的几何含义，它指的是点 $(\alpha,\phi(\alpha))$ 必须在两条直线

$$l_1(\alpha)=\phi(0)+c\alpha\nabla f(x^k)^{\mathrm{T}}d^k,$$
$$l_2(\alpha)=\phi(0)+(1-c)\alpha\nabla f(x^k)^{\mathrm{T}}d^k$$

之间．如图5.2所示，区间 $[\alpha_1,\alpha_2]$ 中的点均满足 Goldstein 准则．同时我们也注意到 Goldstein 准则确实去掉了过小的 α.

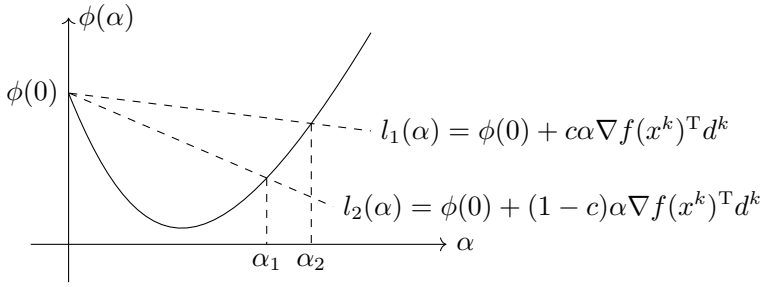


图 5.2 Goldstein 准则

3. Wolfe 准则

Goldstein 准则能够使得函数值充分下降，但是它可能避开了最优的函数值. 如图5.2所示，一维函数 $\phi(\alpha)$ 的最小值点并不在满足 Goldstein 准则的区间 $[\alpha_1,\alpha_2]$ 中. 为此我们引入 Armijo-Wolfe 准则，简称 Wolfe 准则.

定义 5.3 (Wolfe 准则) 设 d^k 是点 x^k 处的下降方向，若

$$f(x^k+\alpha d^k)\leqslant f(x^k)+c_1\alpha\nabla f(x^k)^{\mathrm{T}}d^k, \tag{5.1.4a}$$

$$\nabla f(x^k+\alpha d^k)^{\mathrm{T}}d^k\geqslant c_2\nabla f(x^k)^{\mathrm{T}}d^k, \tag{5.1.4b}$$

则称步长 α 满足 **Wolfe 准则**，其中 $c_1,c_2\in(0,1)$ 为给定的常数且 $c_1<c_2$.

在准则(5.1.4)中，第一个不等式(5.1.4a)即是 Armijo 准则，而第二个不等式(5.1.4b) 则是 Wolfe 准则的本质要求. 注意到 $\nabla f(x^k+\alpha d^k)^{\mathrm{T}}d^k$ 恰好就是 $\phi(\alpha)$ 的导数，Wolfe 准则实际要求 $\phi(\alpha)$ 在点 α 处切线的斜率不能小于 $\phi'(0)$ 的 c_2 倍. 如图5.3所示，在区间 $[\alpha_1,\alpha_2]$ 中的点均满足 Wolfe 准则. 注意到在 $\phi(\alpha)$ 的极小值点 α^* 处有 $\phi'(\alpha^*)=\nabla f(x^k+\alpha^*d^k)^{\mathrm{T}}d^k=0$，因此 α^* 永远满足条件(5.1.4b). 而选择较小的 c_1 可使得 α^* 同时满足条件(5.1.4a)，即 Wolfe 准则在绝大多数情况下会包含线搜索子问题的精确解. 在实际应用中，参数 c_2 通常取为 0.9.

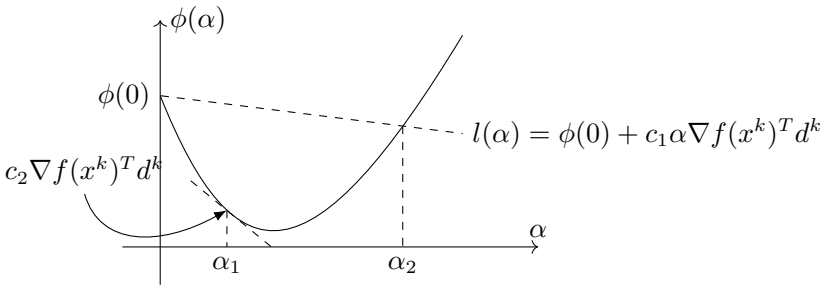


图 5.3 Wolfe 准则

4. 非单调线搜索准则

以上介绍的三种准则都有一个共同点：使用这些准则产生的迭代点列都是单调的. 在实际应用中，非单调算法有时会有更好的效果. 这就需要我们应用非单调线搜索准则，这里介绍其中两种.

定义 5.4 (Grippo [61]) 设 d^k 是点 x^k 处的下降方向，$M>0$ 为给定的正整数. 以下不等式可作为一种线搜索准则：

$$f(x^k+\alpha d^k)\leqslant \max_{0\leqslant j\leqslant \min\{k,M\}} f(x^{k-j})+c_1\alpha\nabla f(x^k)^{\mathrm{T}}d^k, \tag{5.1.5}$$

其中 $c_1\in(0,1)$ 为给定的常数.

准则(5.1.5)和 Armijo 准则非常相似，区别在于 Armijo 准则要求下一次迭代的函数值 $f(x^{k+1})$ 相对于本次迭代的函数值 $f(x^k)$ 有充分下降，而准则(5.1.5) 只需要下一步函数值相比前面至多 M 步以内迭代的函数值有下降就可以了. 显然这一准则的要求比 Armijo 准则更宽，它也不要求 $f(x^k)$ 的单调性.

另一种非单调线搜索准则的定义更加宽泛.

定义 5.5 (Zhang, Hager [137]) 设 d^k 是点 x^k 处的下降方向，以下不等式可作为一种线搜索准则：

$$f(x^k+\alpha d^k)\leqslant C^k+c_1\alpha\nabla f(x^k)^{\mathrm{T}}d^k, \tag{5.1.6}$$

其中 C^k 满足递推式 $C^0=f(x^0), C^{k+1}=\dfrac{1}{Q^{k+1}}(\eta Q^k C^k+f(x^{k+1}))$，序列 $\{Q^k\}$ 满足 $Q^0=1, Q^{k+1}=\eta Q^k+1$，参数 $\eta, c_1\in(0,1)$.

我们可以用以下的方式理解这个准则：变量 C^k 实际上是本次搜索准则的参照函数值，即充分下降性质的起始标准；而下一步的标准 C^{k+1} 则是函数值 $f(x^{k+1})$ 和 C^k 的凸组合，并非仅仅依赖于 $f(x^{k+1})$，而凸组合的两个系数由参数 η 决定. 可以看到当 $\eta=0$ 时，此准则就是 Armijo 准则.

5.1.2 线搜索算法

本小节介绍在实际中使用的线搜索算法. 之前的讨论已经初步介绍了回退法（算法5.1），并指出该算法可以用于寻找 Armijo 准则(5.1.2)的步长. 实际上只要修改一下算法的终止条件，回退法就可以被用在其他线搜索准则之上，例如之前我们提到的两种非单调线搜索准则(5.1.5)和(5.1.6). 回退法的实现简单、原理直观，所以它是最常用的线搜索算法之一. 然而，回退法的缺点也很明显：第一，它无法保证找到满足 Wolfe 准则的步长，即条件(5.1.4b)不一定成立，但对一些优化算法而言，找到满足 Wolfe 准则的步长是十分必要的；第二，回退法以指数的方式缩小步长，因此对初值 $\hat{\alpha}$ 和参

数 γ 的选取比较敏感，当 γ 过大时每一步试探步长改变量很小，此时回退法效率比较低，当 γ 过小时回退法过于激进，导致最终找到的步长太小，错过了选取大步长的机会. 下面简单介绍其他类型的线搜索算法.

为了提高回退法的效率，我们有基于多项式插值的线搜索算法. 假设初始步长 $\hat{\alpha}_0$ 已给定，如果经过验证，$\hat{\alpha}_0$ 不满足 Armijo 准则，下一步就需要减小试探步长. 和回退法不同，我们不直接将 $\hat{\alpha}_0$ 缩小常数倍，而是基于 $\phi(0), \phi'(0), \phi(\hat{\alpha}_0)$ 这三个信息构造一个二次插值函数 $p_2(\alpha)$，即寻找二次函数 $p_2(\alpha)$ 满足

$$p_2(0) = \phi(0), \quad p_2'(0) = \phi'(0), \quad p_2(\hat{\alpha}_0) = \phi(\hat{\alpha}_0).$$

由于二次函数只有三个参数，以上三个条件可以唯一决定 $p_2(\alpha)$，而且不难验证 $p_2(\alpha)$ 的最小值点恰好位于 $(0, \hat{\alpha}_0)$ 内. 此时取 $p_2(\alpha)$ 的最小值点 $\hat{\alpha}_1$ 作为下一个试探点，利用同样的方式不断递归下去直至找到满足 Armijo 准则的点.

基于插值的线搜索算法可以有效减少试探次数，但仍然不能保证找到的步长满足 Wolfe 准则. 为此，Fletcher 提出了一个用于寻找满足 Wolfe 准则的算法 [45]. 这个算法比较复杂，有较多细节，这里不展开叙述，读者可以参考 [45,98].

5.1.3 收敛性分析

这一小节给出使用不同线搜索准则导出的算法的收敛性. 此收敛性建立在一般的线搜索类算法的框架上，因此得到的结论也比较弱. 不过它可以帮助我们理解线搜索类算法收敛的本质要求.

定理 5.1 (Zoutendijk) *考虑一般的迭代格式(5.1.1)，其中 d^k 是搜索方向，α_k 是步长，且在迭代过程中 Wolfe 准则 (5.1.4)满足. 假设目标函数 f 下有界、连续可微且梯度 L-利普希茨连续，即*

$$\|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\|, \quad \forall\, x, y \in \mathbb{R}^n,$$

那么

$$\sum_{k=0}^{\infty} \cos^2\theta_k \|\nabla f(x^k)\|^2 < +\infty, \tag{5.1.7}$$

其中 $\cos\theta_k$ 为负梯度 $-\nabla f(x^k)$ 和下降方向 d^k 夹角的余弦，即

$$\cos\theta_k = \frac{-\nabla f(x^k)^{\mathrm{T}} d^k}{\|\nabla f(x^k)\| \|d^k\|}.$$

不等式(5.1.7)也被称为 **Zoutendijk 条件**.

证明. 由条件(5.1.4b),

$$\Big(\nabla f(x^{k+1}) - \nabla f(x^k)\Big)^{\mathrm{T}} d^k \geqslant (c_2 - 1)\nabla f(x^k)^{\mathrm{T}} d^k.$$

由柯西不等式和梯度 L-利普希茨连续性质,

$$\Big(\nabla f(x^{k+1}) - \nabla f(x^k)\Big)^{\mathrm{T}} d^k \leqslant \|\nabla f(x^{k+1}) - \nabla f(x^k)\|\|d^k\| \leqslant \alpha_k L\|d^k\|^2.$$

结合上述两式可得

$$\alpha_k \geqslant \frac{c_2 - 1}{L}\frac{\nabla f(x^k)^{\mathrm{T}} d^k}{\|d^k\|^2}.$$

注意到 $\nabla f(x^k)^{\mathrm{T}} d^k < 0$, 将上式代入条件(5.1.4a), 则

$$f(x^{k+1}) \leqslant f(x^k) + c_1\frac{c_2 - 1}{L}\frac{\left(\nabla f(x^k)^{\mathrm{T}} d^k\right)^2}{\|d^k\|^2}.$$

根据 θ_k 的定义, 此不等式可等价表述为

$$f(x^{k+1}) \leqslant f(x^k) + c_1\frac{c_2 - 1}{L}\cos^2\theta_k\|\nabla f(x^k)\|^2.$$

再关于 k 求和, 我们有

$$f(x^{k+1}) \leqslant f(x^0) - c_1\frac{1 - c_2}{L}\sum_{j=0}^{k}\cos^2\theta_j\|\nabla f(x^j)\|^2.$$

又因为函数 f 是下有界的, 且由 $0 < c_1 < c_2 < 1$ 可知 $c_1(1 - c_2) > 0$, 因此当 $k \to \infty$ 时,

$$\sum_{j=0}^{\infty}\cos^2\theta_j\|\nabla f(x^j)\|^2 < +\infty. \qquad \square$$

定理 5.1 指出, 只要迭代点满足 Wolfe 准则, 对梯度利普希茨连续且下有界函数总能推出(5.1.7)式成立. 实际上采用 Goldstein 准则也可推出类似的条件. Zoutendijk 定理刻画了线搜索准则的性质, 配合下降方向 d^k 的选取方式我们可以得到最基本的收敛性.

推论 5.1 (线搜索算法的收敛性) 对于迭代法(5.1.1), 设 θ_k 为每一步负梯度 $-\nabla f(x^k)$ 与下降方向 d^k 的夹角, 并假设对任意的 k, 存在常数 $\gamma > 0$, 使得

$$\theta_k < \frac{\pi}{2} - \gamma, \tag{5.1.8}$$

则在定理5.1成立的条件下，有

$$\lim_{k\to\infty} \nabla f(x^k) = 0.$$

证明. 假设结论不成立，即存在子列 $\{k_l\}$ 和正常数 $\delta > 0$，使得

$$\|\nabla f(x^{k_l})\| \geqslant \delta, \quad l = 1,2,\cdots.$$

根据 θ_k 的假设，对任意的 k，

$$\cos\theta_k > \sin\gamma > 0.$$

我们仅考虑和式(5.1.7)的第 k_l 项，有

$$\begin{aligned}\sum_{k=0}^{\infty}\cos^2\theta_k\|\nabla f(x^k)\|^2 &\geqslant \sum_{l=1}^{\infty}\cos^2\theta_{k_l}\|\nabla f(x^{k_l})\|^2\\ &\geqslant \sum_{l=1}^{\infty}(\sin^2\gamma)\cdot\delta^2 \to +\infty,\end{aligned}$$

这显然和定理5.1矛盾. 因此必有

$$\lim_{k\to\infty} \nabla f(x^k) = 0. \qquad \square$$

推论5.1建立在 Zoutendijk 条件之上，它的本质要求是关系(5.1.8)，即每一步的下降方向 d^k 和负梯度方向不能趋于正交. 这个条件的几何直观明显：当下降方向 d^k 和梯度正交时，根据泰勒展开的一阶近似，目标函数值 $f(x^k)$ 几乎不发生改变. 因此我们要求 d^k 与梯度正交方向夹角有一致的下界. 后面会介绍多种 d^k 的选取方法，在选取 d^k 时条件(5.1.8)总得到满足.

总的来说，推论5.1仅仅给出了最基本的收敛性，而没有更进一步回答算法的收敛速度. 这是由于算法收敛速度极大地取决于 d^k 的选取. 接下来我们将着重介绍如何选取下降方向 d^k.

5.2 梯度类算法

本节介绍梯度类算法，其本质是仅仅使用函数的一阶导数信息选取下降方向 d^k. 这其中最基本的算法是梯度下降法，即直接选择负梯度作为下降方向 d^k. 梯度下降法的方向选取非常直观，实际应用范围非常广，因此它在优化算法中的地位可相当于高斯消元法在线性方程组算法中的地位. 此外我们也会介绍 BB 方法. 该方法作为一种梯度法的变形，虽然理论性质目前仍不完整，但由于它有优秀的数值表现，也是在实际应用中使用较多的一种算法.

5.2.1 梯度下降法

对于光滑函数 $f(x)$，在迭代点 x^k 处，我们需要选择一个较为合理的 d^k 作为下降方向. 注意到 $\phi(\alpha) = f(x^k + \alpha d^k)$ 有泰勒展开

$$\phi(\alpha) = f(x^k) + \alpha \nabla f(x^k)^{\mathrm{T}} d^k + \mathcal{O}(\alpha^2 \|d^k\|^2),$$

根据柯西不等式，当 α 足够小时，取 $d^k = -\nabla f(x^k)$ 会使得函数下降最快. 因此梯度法就是选取 $d^k = -\nabla f(x^k)$ 的算法，它的迭代格式为

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k). \tag{5.2.1}$$

步长 α_k 的选取可依赖于上一节的线搜索算法，也可直接选取固定的 α_k.

为了直观地理解梯度法的迭代过程，我们以二次函数为例来展示该过程.

例 5.2 (二次函数的梯度法) 设二次函数 $f(x,y) = x^2 + 10y^2$，初始点 (x^0, y^0) 取为 $(10,1)$，取固定步长 $\alpha_k = 0.085$. 我们使用梯度法(5.2.1)进行 15 次迭代，结果如图5.4所示.

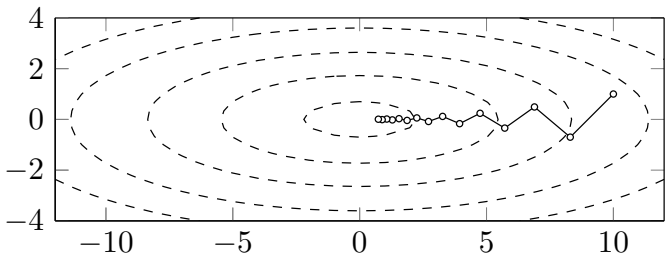


图 5.4 梯度法的前 15 次迭代

实际上，对正定二次函数有如下收敛定理（证明见 [86]）:

定理 5.2 考虑正定二次函数

$$f(x) = \frac{1}{2} x^{\mathrm{T}} A x - b^{\mathrm{T}} x,$$

其最优值点为 x^*. 若使用梯度法(5.2.1)并选取 α_k 为精确线搜索步长，即

$$\alpha_k = \frac{\|\nabla f(x^k)\|^2}{\nabla f(x^k)^{\mathrm{T}} A \nabla f(x^k)}, \tag{5.2.2}$$

则梯度法关于迭代点列 $\{x^k\}$ 是 Q-线性收敛的，即

$$\|x^{k+1}-x^*\|_A^2 \leqslant \left(\frac{\lambda_1-\lambda_n}{\lambda_1+\lambda_n}\right)^2 \|x^k-x^*\|_A^2,$$

其中 λ_1,λ_n 分别为 A 的最大、最小特征值，$\|x\|_A \xlongequal{\text{def}} \sqrt{x^{\mathrm{T}}Ax}$ 为由正定矩阵 A 诱导的范数.

定理5.2指出使用精确线搜索的梯度法在正定二次问题上有 Q-线性收敛速度. 线性收敛速度的常数和矩阵 A 最大特征值与最小特征值之比有关. 从等高线角度来看，这个比例越大则 $f(x)$ 的等高线越扁平，图5.4中迭代路径折返频率会随之变高，梯度法收敛也就越慢. 这个结果其实说明了梯度法的一个很重大的缺陷：当目标函数的海瑟矩阵条件数较大时，它的收敛速度会非常缓慢.

接下来我们介绍当 $f(x)$ 为梯度利普希茨连续的凸函数时，梯度法(5.2.1)的收敛性质.

定理 5.3 (梯度法在凸函数上的收敛性) 设函数 $f(x)$ 为凸的梯度 L-利普希茨连续函数，$f^*=f(x^*)=\inf\limits_x f(x)$ 存在且可达. 如果步长 α_k 取为常数 α 且满足 $0<\alpha<\dfrac{1}{L}$，那么由迭代 (5.2.1) 得到的点列 $\{x^k\}$ 的函数值收敛到最优值，且在函数值的意义下收敛速度为 $\mathcal{O}\left(\dfrac{1}{k}\right)$.

证明. 因为函数 f 是利普希茨可微函数，对任意的 x，根据(2.2.3)式，

$$f(x-\alpha\nabla f(x)) \leqslant f(x)-\alpha\left(1-\frac{L\alpha}{2}\right)\|\nabla f(x)\|^2. \tag{5.2.3}$$

现在记 $\tilde{x}=x-\alpha\nabla f(x)$ 并限制 $0<\alpha<\dfrac{1}{L}$，我们有

$$\begin{aligned} f(\tilde{x}) &\leqslant f(x)-\frac{\alpha}{2}\|\nabla f(x)\|^2 \\ &\leqslant f^*+\nabla f(x)^{\mathrm{T}}(x-x^*)-\frac{\alpha}{2}\|\nabla f(x)\|^2 \\ &= f^*+\frac{1}{2\alpha}\left(\|x-x^*\|^2-\|x-x^*-\alpha\nabla f(x)\|^2\right) \\ &= f^*+\frac{1}{2\alpha}(\|x-x^*\|^2-\|\tilde{x}-x^*\|^2), \end{aligned}$$

其中第一个不等式是由于(5.2.3)式，第二个不等式为 f 的凸性. 在上式中取

$x = x^{i-1}, \tilde{x} = x^i$ 并将不等式对 $i = 1,2,\cdots,k$ 求和得到

$$\begin{aligned}\sum_{i=1}^{k}(f(x^i) - f^*) &\leqslant \frac{1}{2\alpha}\sum_{i=1}^{k}\left(\|x^{i-1} - x^*\|^2 - \|x^i - x^*\|^2\right)\\ &= \frac{1}{2\alpha}\left(\|x^0 - x^*\|^2 - \|x^k - x^*\|^2\right)\\ &\leqslant \frac{1}{2\alpha}\|x^0 - x^*\|^2.\end{aligned}$$

根据(5.2.3)式得知 $f(x^i)$ 是非增的，所以

$$f(x^k) - f^* \leqslant \frac{1}{k}\sum_{i=1}^{k}(f(x^i) - f^*) \leqslant \frac{1}{2k\alpha}\|x^0 - x^*\|^2. \qquad \square$$

如果函数 f 还是 m-强凸函数，则梯度法的收敛速度会进一步提升为Q-线性收敛.

在给出收敛性证明之前，我们需要以下的引理来揭示凸的梯度 L-利普希茨连续函数的另一个重要性质.

引理 5.1 *设函数 $f(x)$ 是 $\mathbb{R}^n$ 上的凸可微函数，则以下结论等价：*

(1) f 的梯度为 L-利普希茨连续的；

(2) 函数 $g(x) \overset{\text{def}}{=\!=} \dfrac{L}{2}x^{\mathrm{T}}x - f(x)$ 是凸函数；

*(3) $\nabla f(x)$ 有**余强制性**，即对任意的 $x,y \in \mathbb{R}^n$，有*

$$(\nabla f(x) - \nabla f(y))^{\mathrm{T}}(x - y) \geqslant \frac{1}{L}\|\nabla f(x) - \nabla f(y)\|^2. \tag{5.2.4}$$

证明. (1) $\Longrightarrow$ (2) 即证 $g(x)$ 的单调性. 对任意 $x,y \in \mathbb{R}^n$，

$$\begin{aligned}(\nabla g(x) - \nabla g(y))^{\mathrm{T}}(x - y) &= L\|x - y\|^2 - (\nabla f(x) - \nabla f(y))^{\mathrm{T}}(x - y)\\ &\geqslant L\|x - y\|^2 - \|x - y\|\|\nabla f(x) - \nabla f(y)\| \geqslant 0.\end{aligned}$$

因此 $g(x)$ 为凸函数.

(2) $\Longrightarrow$ (3) 构造辅助函数

$$\begin{aligned}f_x(z) &= f(z) - \nabla f(x)^{\mathrm{T}}z,\\ f_y(z) &= f(z) - \nabla f(y)^{\mathrm{T}}z,\end{aligned}$$

容易验证 f_x 和 f_y 均为凸函数. 根据已知条件, $g_x(z) = \frac{L}{2}z^{\mathrm{T}}z - f_x(z)$ 关于 z 是凸函数. 根据凸函数的性质, 我们有

$$g_x(z_2) \geqslant g_x(z_1) + \nabla g_x(z_1)^{\mathrm{T}}(z_2 - z_1), \quad \forall\, z_1, z_2 \in \mathbb{R}^n.$$

整理可推出 $f_x(z)$ 有二次上界, 且对应的系数也为 L. 注意到 $\nabla f_x(x) = 0$, 这说明 x 是 $f_x(z)$ 的最小值点. 再由推论 2.1 的证明过程,

$$\begin{aligned} f_x(y) - f_x(x) &= f(y) - f(x) - \nabla f(x)^{\mathrm{T}}(y - x) \\ &\geqslant \frac{1}{2L}\|\nabla f_x(y)\|^2 = \frac{1}{2L}\|\nabla f(y) - \nabla f(x)\|^2. \end{aligned}$$

同理, 对 $f_y(z)$ 进行类似的分析可得

$$f(x) - f(y) - \nabla f(y)^{\mathrm{T}}(x - y) \geqslant \frac{1}{2L}\|\nabla f(y) - \nabla f(x)\|^2.$$

将以上两式不等号左右分别相加, 可得余强制性(5.2.4).

(3) $\Longrightarrow$ (1) 由余强制性和柯西不等式,

$$\begin{aligned} \frac{1}{L}\|\nabla f(x) - \nabla f(y)\|^2 &\leqslant (\nabla f(x) - \nabla f(y))^{\mathrm{T}}(x - y) \\ &\leqslant \|\nabla f(x) - \nabla f(y)\|\|x - y\|, \end{aligned}$$

整理后即可得到 $f(x)$ 是梯度 L-利普希茨连续的. □

引理 5.1 说明在 f 为凸函数的条件下, 梯度 L-利普希茨连续、二次上界、余强制性三者是等价的, 知道其中一个性质就可推出剩下两个. 接下来给出梯度法在强凸函数下的收敛性.

定理 5.4 (梯度法在强凸函数上的收敛性) *设函数 $f(x)$ 为 m-强凸的梯度 L-利普希茨连续函数, $f^* = f(x^*) = \inf\limits_x f(x)$ 存在且可达. 如果步长 α 满足 $0 < \alpha \leqslant \frac{2}{m+L}$, 那么由梯度下降法(5.2.1)迭代得到的点列 $\{x^k\}$ 收敛到 x^*, 且为 Q-线性收敛.*

证明. 首先根据 f 强凸且 ∇f 利普希茨连续, 可得

$$g(x) = f(x) - \frac{m}{2}x^{\mathrm{T}}x$$

为凸函数且 $\frac{L-m}{2}x^{\mathrm{T}}x - g(x)$ 为凸函数. 由引理5.1 可得 $g(x)$ 是梯度 $(L-m)$-利普希茨连续的. 再次利用引理5.1可得关于 $g(x)$ 的余强制性

$$(\nabla g(x) - \nabla g(y))^{\mathrm{T}}(x - y) \geqslant \frac{1}{L-m}\|\nabla g(x) - \nabla g(y)\|^2. \tag{5.2.5}$$

代入 $g(x)$ 的表达式，可得

$$\begin{aligned}&(\nabla f(x)-\nabla f(y))^{\mathrm{T}}(x-y)\\ \geqslant&\frac{mL}{m+L}\|x-y\|^2+\frac{1}{m+L}\|\nabla f(x)-\nabla f(y)\|^2.\end{aligned}\tag{5.2.6}$$

然后我们估计在固定步长下梯度法的收敛速度. 设步长 $\alpha\in\left(0,\dfrac{2}{m+L}\right]$，则

$$\begin{aligned}\|x^{k+1}-x^*\|^2&=\|x^k-\alpha\nabla f(x^k)-x^*\|^2\\ &=\|x^k-x^*\|^2-2\alpha\nabla f(x^k)^{\mathrm{T}}(x^k-x^*)+\alpha^2\|\nabla f(x^k)\|^2\\ &\leqslant\left(1-\alpha\frac{2mL}{m+L}\right)\|x^k-x^*\|^2+\alpha\left(\alpha-\frac{2}{m+L}\right)\|\nabla f(x^k)\|^2\\ &\leqslant\left(1-\alpha\frac{2mL}{m+L}\right)\|x^k-x^*\|^2.\end{aligned}$$

其中第一个不等式是对 x^k,x^* 应用(5.2.6)式并注意到 $\nabla f(x^*)=0$. 因此，

$$\|x^k-x^*\|^2\leqslant c^k\|x^0-x^*\|^2,\quad c=1-\alpha\frac{2mL}{m+L}<1,$$

即在强凸函数的条件下，梯度法是 Q-线性收敛的. □

5.2.2 Barzilai-Borwein 方法

由上一小节可以知道，当问题的条件数很大，也即问题比较病态时，梯度下降法的收敛性质会受到很大影响. Barzilai-Borwein (BB) 方法是一种特殊的梯度法，经常比一般的梯度法有着更好的效果. 从形式上来看，BB 方法的下降方向仍是点 x^k 处的负梯度方向 $-\nabla f(x^k)$，但步长 α_k 并不是直接由线搜索算法给出的. 考虑梯度下降法的格式:

$$x^{k+1}=x^k-\alpha_k\nabla f(x^k),$$

这种格式也可以写成

$$x^{k+1}=x^k-D^k\nabla f(x^k),$$

其中 $D^k=\alpha_k I$. BB 方法选取的 α_k 是如下两个最优问题之一的解:

$$\min_{\alpha}\quad\|\alpha y^{k-1}-s^{k-1}\|^2,\tag{5.2.7}$$

$$\min_{\alpha}\quad\|y^{k-1}-\alpha^{-1}s^{k-1}\|^2,\tag{5.2.8}$$

其中引入记号 $s^{k-1} \xlongequal{\text{def}} x^k - x^{k-1}$ 以及 $y^{k-1} \xlongequal{\text{def}} \nabla f(x^k) - \nabla f(x^{k-1})$. 在这里先直接写出问题(5.2.7)和(5.2.8)，它们的实际含义将在第5.5 小节中给出合理的解释.

容易验证问题(5.2.7)和(5.2.8)的解分别为

$$\alpha_{\mathrm{BB1}}^k \xlongequal{\text{def}} \frac{(s^{k-1})^{\mathrm{T}} y^{k-1}}{(y^{k-1})^{\mathrm{T}} y^{k-1}} \quad \text{和} \quad \alpha_{\mathrm{BB2}}^k \xlongequal{\text{def}} \frac{(s^{k-1})^{\mathrm{T}} s^{k-1}}{(s^{k-1})^{\mathrm{T}} y^{k-1}}, \tag{5.2.9}$$

因此可以得到 BB 方法的两种迭代格式：

$$x^{k+1} = x^k - \alpha_{\mathrm{BB1}}^k \nabla f(x^k), \tag{5.2.10a}$$

$$x^{k+1} = x^k - \alpha_{\mathrm{BB2}}^k \nabla f(x^k). \tag{5.2.10b}$$

我们从表达式(5.2.9)注意到，计算两种 BB 步长的任何一种仅仅需要函数相邻两步的梯度信息和迭代点信息，不需要任何线搜索算法即可选取算法步长. 因为这个特点，BB 方法的使用范围特别广泛. 对于一般的问题，通过(5.2.9)式计算出的步长可能过大或过小，因此我们还需要将步长做上界和下界的截断，即选取 $0 < \alpha_m < \alpha_M$ 使得

$$\alpha_m \leqslant \alpha_k \leqslant \alpha_M.$$

还需注意的是，BB 方法本身是非单调方法，有时也配合非单调收敛准则使用以获得更好的实际效果. 算法5.2给出了一种 BB 方法的框架.

算法 5.2 非单调线搜索的 BB 方法

1. 给定 x^0，选取初值 $\alpha > 0$，整数 $M \geqslant 0$，$c_1, \beta, \varepsilon \in (0,1)$，$k = 0$.
2. **while** $\|\nabla f(x^k)\| > \varepsilon$ **do**
3. **while** $f(x^k - \alpha \nabla f(x^k)) \geqslant \max\limits_{0 \leqslant j \leqslant \min(k,M)} f(x^{k-j}) - c_1 \alpha \|\nabla f(x^k)\|^2$ **do**
4. 令 $\alpha \leftarrow \beta \alpha$.
5. **end while**
6. 令 $x^{k+1} = x^k - \alpha \nabla f(x^k)$.
7. 根据公式(5.2.9)之一计算 α，并做截断使得 $\alpha \in [\alpha_m, \alpha_M]$.
8. $k \leftarrow k + 1$.
9. **end while**

我们仍然使用例 5.2 来说明 BB 方法的迭代过程.

例 5.3 (二次函数的 BB 方法) 设二次函数 $f(x,y)=x^2+10y^2$，并使用 BB 方法进行迭代，初始点为 $(-10,-1)$，结果如图5.5所示．为了方便对比，我们也在此图中描绘了梯度法的迭代过程．可以很明显看出 BB 方法的收敛速度较快，在经历 15 次迭代后已经接近最优值点．从等高线也可观察到 BB 方法是非单调方法．

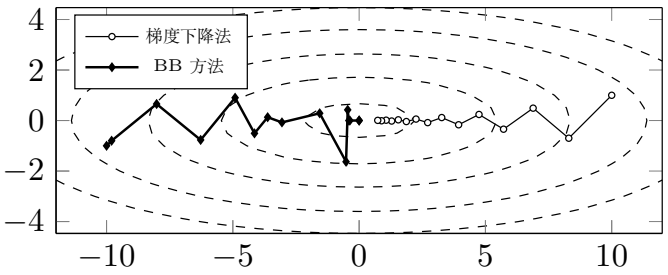


图 5.5 梯度法与 BB 方法的前 15 次迭代

实际上，对于正定二次函数，BB 方法有 R-线性收敛速度．对于一般问题，BB 方法的收敛性还需要进一步研究．但即便如此，使用 BB 方法的步长通常都会减少算法的迭代次数．因此在编写算法时，选取 BB 方法的步长是常用加速策略之一．

5.2.3 应用举例

1. LASSO 问题求解

本小节利用梯度法来求解 LASSO 问题．这个问题的原始形式为

$$\min \quad f(x)=\frac{1}{2}\|Ax-b\|^2+\mu\|x\|_1.$$

LASSO 问题的目标函数 $f(x)$ 不光滑，在某些点处无法求出梯度，因此不能直接对原始问题使用梯度法求解．考虑到目标函数的不光滑项为 $\|x\|_1$，它实际上是 x 各个分量绝对值的和，如果能找到一个光滑函数来近似绝对值函数，那么梯度法就可以被用在 LASSO 问题的求解上．在实际应用中，我

们可以考虑如下一维光滑函数：

$$l_\delta(x) = \begin{cases} \dfrac{1}{2\delta}x^2, & |x| < \delta, \\ |x| - \dfrac{\delta}{2}, & \text{其他}. \end{cases} \tag{5.2.11}$$

定义(5.2.11)实际上是 Huber 损失函数的一种变形，当 $\delta \to 0$ 时，光滑函数 $l_\delta(x)$ 和绝对值函数 $|x|$ 会越来越接近．图5.6展示了当 δ 取不同值时 $l_\delta(x)$ 的图形.

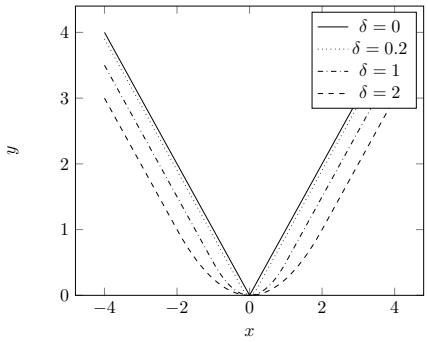


图 5.6 当 δ 取不同值时 $l_\delta(x)$ 的图形

因此，我们构造光滑化 LASSO 问题为

$$\min \quad f_\delta(x) = \frac{1}{2}\|Ax - b\|^2 + \mu L_\delta(x), \tag{5.2.12}$$

其中 δ 为给定的光滑化参数，在这里

$$L_\delta(x) = \sum_{i=1}^{n} l_\delta(x_i),$$

即对 x 的每个分量作用光滑函数 (5.2.11) 再整体求和．容易计算出 $f_\delta(x)$ 的梯度为

$$\nabla f_\delta(x) = A^{\mathrm{T}}(Ax - b) + \mu \nabla L_\delta(x),$$

其中 $\nabla L_\delta(x)$ 是逐个分量定义的：

$$(\nabla L_\delta(x))_i = \begin{cases} \operatorname{sign}(x_i), & |x_i| > \delta, \\ \dfrac{x_i}{\delta}, & |x_i| \leqslant \delta. \end{cases}$$

显然 $f_\delta(x)$ 的梯度是利普希茨连续的，且相应常数为 $L = \|A^{\mathrm{T}}A\|_2 + \dfrac{\mu}{\delta}$. 根据定理5.3，固定步长需不超过 $\dfrac{1}{L}$ 才能保证算法收敛，如果 δ 过小，那么我们需要选取充分小的步长 α_k 使得梯度法收敛.

图 5.7展示了光滑化 LASSO 问题的求解结果. 在 MATLAB 环境中，我们用如下方式生成 A, b:

```
m = 512; n = 1024;
A = randn(m, n);
u = sprandn(n, 1, r);
b = A * u;
```

其中 r 用来控制真解 u 的稀疏度（u 中非零元个数与总元素个数的比值为 r）. 这里取稀疏度 $r = 0.1$，正则化参数 $\mu = 10^{-3}$. 只要

$$|f_\delta(x^k) - f_\delta(x^{k-1})| < 10^{-8}, \text{ 或者 } \|\nabla f_\delta(x)\| < 10^{-6}$$

或者最大迭代步数达到 3000，则算法停止. 为了加快算法的收敛速度，可以采用连续化策略来从较大的正则化参数 μ_0 逐渐减小到 μ. 具体地，对于每一个 μ_t，我们调用带 BB 步长的光滑化梯度法（这里光滑化参数 $\delta_t = 10^{-3}\mu_t$）来求解对应的子问题. 每个子问题的终止条件设为

$$|f_\delta(x^k) - f_\delta(x^{k-1})| < 10^{-4-t}, \text{ 或者 } \|\nabla f_\delta(x)\| < 10^{-1-t}.$$

当 μ_t 的子问题求解完之后，设置

$$\mu_{t+1} = \max\{\mu_t\eta, \mu\}, \tag{5.2.13}$$

其中 η 为缩小因子，这里取为 0.1. 第6.1.4小节将给出连续化策略合理性的解释.

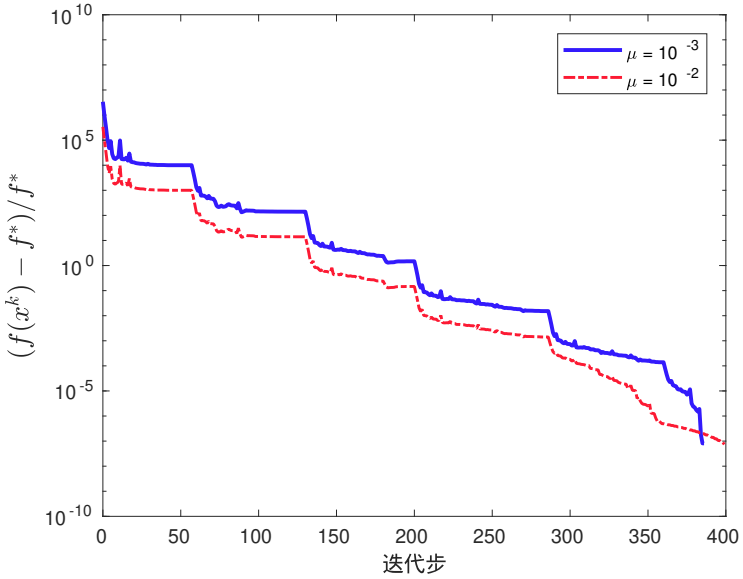
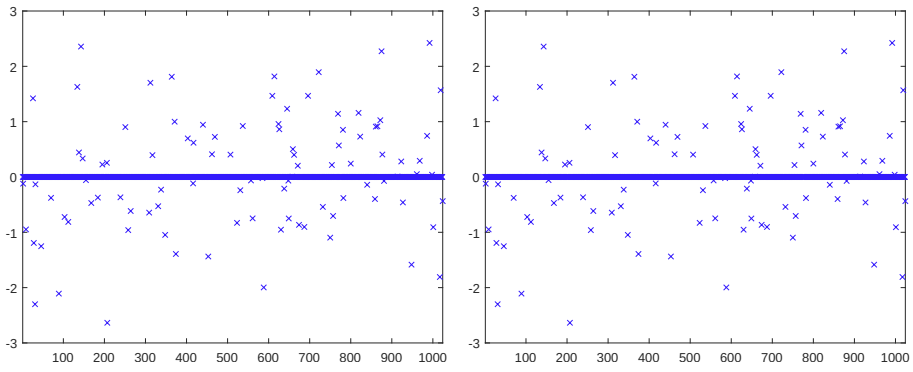


图 5.7 光滑化 LASSO 问题求解迭代过程



(a) 精确解 (b) 梯度法解

图 5.8 光滑化 LASSO 问题求解结果

可以看到，在连续化策略的帮助下，光滑化梯度法在 400 步左右收敛到 LASSO 问题的解.

5.3 次梯度算法

上一节讨论了梯度下降法，使用该方法的前提为目标函数 $f(x)$ 是一阶可微的. 在实际应用中经常会遇到不可微的函数，对于这类函数我们无法在每个点处求出梯度，但往往它们的最优值都是在不可微点处取到的. 为了能处理这种情形，这一节介绍次梯度算法.

5.3.1 次梯度算法结构

现在我们在问题(5.0.1)中假设 $f(x)$ 为凸函数，但不一定可微. 对凸函数可以在定义域的内点处定义次梯度 $g \in \partial f(x)$. 类比梯度法的构造，我们有如下次梯度算法的迭代格式：

$$x^{k+1} = x^k - \alpha_k g^k, \quad g^k \in \partial f(x^k), \tag{5.3.1}$$

其中 $\alpha_k > 0$ 为步长. 它通常有如下四种选择：

(1) 固定步长 $\alpha_k = \alpha$；

(2) 固定 $\|x^{k+1} - x^k\|$，即 $\alpha_k \|g^k\|$ 为常数；

(3) 消失步长 $\alpha_k \to 0$ 且 $\sum_{k=0}^{\infty} \alpha_k = +\infty$；

(4) 选取 α_k 使其满足某种线搜索准则.

次梯度算法(5.3.1)的构造虽然是受梯度法(5.2.1)的启发，但在很多方面次梯度算法有其独特性质. 首先，我们知道次微分 $\partial f(x)$ 是一个集合，在次梯度算法的构造中只要求从这个集合中选出一个次梯度即可，但在实际中不同的次梯度取法可能会产生截然不同的效果；其次，对于梯度法，判断一阶最优性条件只需要验证 $\|\nabla f(x^*)\|$ 是否充分小即可，但对于次梯度算法，根据第四章的定理4.4，有 $0 \in \partial f(x^*)$，而这个条件在实际应用中往往是不易直接验证的，这导致我们不能使用定理4.4作为次梯度算法的停机条件；此外，步长选取在次梯度法中的影响非常大，下一小节将讨论在不同步长取法下次梯度算法的收敛性质.

5.3.2 收敛性分析

本小节讨论次梯度算法的收敛性. 首先我们列出 $f(x)$ 所要满足的基本假设.

假设 5.1 对无约束优化问题(5.0.1)，目标函数 $f(x)$ 满足：

(1) f 为凸函数；

(2) f 至少存在一个有限的极小值点 x^*，且 $f(x^*) > -\infty$；

(3) f 的次梯度是有界的，即

$$\|g\| \leqslant G, \quad \forall\, g \in \partial f(x), x \in \mathbb{R}^n_{m_i}.$$

1. 不同步长下的收敛性

对于次梯度算法，一个重要的观察就是它并不是一个下降方法，即无法保证 $f(x^{k+1}) < f(x^k)$，这给收敛性的证明带来了困难. 不过我们可以分析 $f(x)$ 历史迭代的最优点所满足的性质，实际上有如下定理.

定理 5.5 (次梯度算法的收敛性) 在假设5.1的条件下，设 $\{\alpha_k > 0\}$ 为任意步长序列，$\{x^k\}$ 是由算法(5.3.1)产生的迭代序列，则对任意的 $k \geqslant 0$，有

$$2\left(\sum_{i=0}^{k}\alpha_i\right)(\hat{f}^k - f^*) \leqslant \|x^0 - x^*\|^2 + \sum_{i=0}^{k}\alpha_i^2 G^2, \tag{5.3.2}$$

其中 x^* 是 $f(x)$ 的一个全局极小值点，$f^* = f(x^*)$，$\hat{f}^k$ 为前 k 次迭代 $f(x)$ 的最小值，即

$$\hat{f}^k = \min_{0\leqslant i\leqslant k} f(x^i).$$

证明. 该证明的关键是估计迭代点 x^k 与最小值点 x^* 之间的距离满足的关系. 根据迭代格式(5.3.1)，

$$\begin{aligned}\|x^{i+1} - x^*\|^2 &= \|x^i - \alpha_i g^i - x^*\|^2 \\ &= \|x^i - x^*\|^2 - 2\alpha_i\langle g^i, x^i - x^*\rangle + \alpha_i^2\|g^i\|^2 \\ &\leqslant \|x^i - x^*\|^2 - 2\alpha_i(f(x^i) - f^*) + \alpha_i^2 G^2.\end{aligned} \tag{5.3.3}$$

这里最后一个不等式是根据次梯度的定义和 $\|g^i\| \leqslant G$. 将(5.3.3)式移项，等价于

$$2\alpha_i(f(x^i) - f^*) \leqslant \|x^i - x^*\|^2 - \|x^{i+1} - x^*\|^2 + \alpha_i^2 G^2. \tag{5.3.4}$$

对(5.3.4)式两边关于 i 求和（从 0 到 k），有

$$2\sum_{i=0}^{k}\alpha_i(f(x^i)-f^*) \leqslant \|x^0-x^*\|^2-\|x^{k+1}-x^*\|^2+G^2\sum_{i=0}^{k}\alpha_i^2$$
$$\leqslant \|x^0-x^*\|^2+G^2\sum_{i=0}^{k}\alpha_i^2.$$

根据 $\hat{f}^k$ 的定义容易得出

$$\sum_{i=0}^{k}\alpha_i(f(x^i)-f^*) \geqslant \left(\sum_{i=0}^{k}\alpha_i\right)(\hat{f}^k-f^*).$$

结合以上两式可得到结论(5.3.2). □

定理 5.5 揭示了次梯度算法的一些关键性质：次梯度算法的收敛性非常依赖于步长的选取；次梯度算法是非单调算法，可以配套非单调线搜索准则(5.1.5)和 (5.1.6)一起使用. 根据定理 5.5 可以直接得到不同步长取法下次梯度算法的收敛性，证明留给读者完成.

推论 5.2 在假设5.1的条件下，次梯度算法的收敛性满足（$\hat{f}^k$ 的定义和定理5.5中的定义相同）：

(1) 取 $\alpha_i=t$ 为固定步长，则

$$\hat{f}^k-f^* \leqslant \frac{\|x^0-x^*\|^2}{2kt}+\frac{G^2t}{2};$$

(2) 取 α_i 使得 $\|x^{i+1}-x^i\|$ 固定，即 $\alpha_i\|g^i\|=s$ 为常数，则

$$\hat{f}^k-f^* \leqslant \frac{G\|x^0-x^*\|^2}{2ks}+\frac{Gs}{2};$$

(3) 取 α_i 为消失步长，即 $\alpha_i\to 0$ 且 $\sum_{i=0}^{\infty}\alpha_i=+\infty$，则

$$\hat{f}^k-f^* \leqslant \frac{\|x^0-x^*\|^2+G^2\sum_{i=0}^{k}\alpha_i^2}{2\sum_{i=0}^{k}\alpha_i};$$

进一步可得 $\hat{f}^k$ 收敛到 f^*.

从推论5.2可以看到，无论是固定步长还是固定 $\|x^{k+1}-x^k\|$，次梯度算法均没有收敛性，只能收敛到一个次优的解，这和梯度法的结论有很大的不同；只有当 α_k 取消失步长时 $\hat{f}^k$ 才具有收敛性. 一个常用的取法是 $\alpha_k=\dfrac{1}{k}$，这样不但可以保证其为消失步长，还可以保证 $\sum\limits_{i=0}^{\infty}\alpha_i^2$ 有界.

2. 收敛速度和步长的关系

在推论 5.2 中，通过适当选取步长 α_i 可以获得对应次梯度算法的收敛速度. 在这里我们假设 $\|x^0-x^*\|\leqslant R$，即初值和最优解之间的距离有上界. 假设总迭代步数 k 是给定的，根据推论 5.2 的第一个结论，

$$\hat{f}^k-f^*\leqslant\frac{\|x^0-x^*\|^2}{2kt}+\frac{G^2t}{2}\leqslant\frac{R^2}{2kt}+\frac{G^2t}{2}.$$

在固定步长下，由平均值不等式得知当 t 满足

$$\frac{R^2}{2kt}=\frac{G^2t}{2},\quad \text{即 } t=\frac{R}{G\sqrt{k}}$$

时，我们有估计

$$\hat{f}^k-f^*\leqslant\frac{GR}{\sqrt{k}}.$$

以上分析表明要使得目标函数值达到 ε 的精度，即 $\hat{f}^k-f^*\leqslant\varepsilon$，必须取迭代步数 $k=\mathcal{O}\left(\dfrac{1}{\varepsilon^2}\right)$ 且固定步长 α_k 要满足 $t=\mathcal{O}\left(\dfrac{1}{\sqrt{k}}\right)$. 注意这里的固定步长依赖于最大迭代步数，这和之前构造梯度法的步长是不太一样的. 从上面的取法中还可以看出对于满足假设5.1的函数 f，最大迭代步数可以作为判定迭代点是否最优的一个终止准则.

类似地，根据推论 5.2 的第二个结论以及平均值不等式，在固定 $\|x^{i+1}-x^i\|$ 的条件下可以取 $s=\dfrac{R}{\sqrt{k}}$，同样会得到估计

$$\hat{f}^k-f^*\leqslant\frac{GR}{\sqrt{k}}.$$

如果我们知道 $f(x)$ 的更多信息，则可以利用这些信息来选取步长. 例如在某些应用中可预先知道 f^* 的值（但不知道最小值点），根据(5.3.3)式，当

$$\alpha_i=\frac{f(x^i)-f^*}{\|g^i\|^2}$$

时，不等式右侧达到极小，这等价于

$$\frac{(f(x^i)-f^*)^2}{\|g^i\|^2} \leqslant \|x^i-x^*\|^2-\|x^{i+1}-x^*\|^2.$$

递归地利用上式并结合 $\|x^0-x^*\| \leqslant R$ 和 $\|g^i\| \leqslant G$，可以得到

$$\hat{f}^k - f^* \leqslant \frac{GR}{\sqrt{k}}.$$

注意，此时步长的选取已经和最大迭代数无关，它仅仅依赖于当前点处的函数值与最优值的差和次梯度模长.

5.3.3 应用举例

1. LASSO 问题求解

考虑 LASSO 问题

$$\min \quad f(x)=\frac{1}{2}\|Ax-b\|^2+\mu\|x\|_1, \tag{5.3.5}$$

容易得知 $f(x)$ 的一个次梯度为

$$g=A^{\mathrm{T}}(Ax-b)+\mu\mathrm{sign}(x),$$

其中 $\mathrm{sign}(x)$ 是关于 x 逐分量的符号函数. 因此 LASSO 问题的次梯度算法为

$$x^{k+1}=x^k-\alpha_k(A^{\mathrm{T}}(Ax^k-b)+\mu\mathrm{sign}(x^k)),$$

步长 α_k 可选为固定步长或消失步长.

对于 $\mu=10^{-2},10^{-3}$，我们采用连续化次梯度算法进行求解. 停机准则和参数 μ 的连续化设置与第5.2节中的光滑化梯度法一致，且若 $\mu_t>\mu$，则取固定步长 $\dfrac{1}{\lambda_{\max}(A^{\mathrm{T}}A)}$；若 $\mu_t=\mu$，则取步长

$$\frac{1}{\lambda_{\max}(A^{\mathrm{T}}A)\cdot(\max\{k,100\}-99)},$$

其中 k 为迭代步数. 迭代收敛过程见图5.9. 可以看到，次梯度算法在 1000 步左右收敛，比上一节中的光滑化梯度法慢一些.

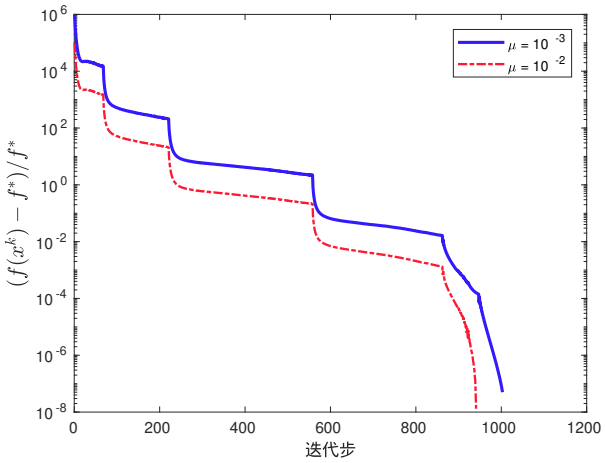


图 5.9 LASSO 问题在不同正则化参数下的求解结果

5.4 牛顿类算法

梯度法仅仅依赖函数值和梯度的信息（即一阶信息），如果函数 $f(x)$ 充分光滑，则可以利用二阶导数信息构造下降方向 d^k. 牛顿类算法就是利用二阶导数信息来构造迭代格式的算法. 由于利用的信息变多，牛顿法的实际表现可以远好于梯度法，但是它对函数 $f(x)$ 的要求也相应变高. 本节首先介绍经典牛顿法的构造和性质，然后介绍一些修正的牛顿法和实际应用.

5.4.1 经典牛顿法

对二次连续可微函数 $f(x)$，考虑 $f(x)$ 在迭代点 x^k 处的二阶泰勒展开

$$f(x^k+d^k)=f(x^k)+\nabla f(x^k)^{\mathrm{T}}d^k+\frac{1}{2}(d^k)^{\mathrm{T}}\nabla^2 f(x^k)d^k+o(\|d^k\|^2). \quad (5.4.1)$$

我们的目的是根据这个二阶近似来选取合适的下降方向 d^k.如果忽略(5.4.1)式中的高阶项，并将等式右边看成关于 d^k 的函数求其稳定点，可以得到

$$\nabla^2 f(x^k)d^k=-\nabla f(x^k). \quad (5.4.2)$$

方程(5.4.2)也被称为**牛顿方程**，容易得出当 $\nabla^2 f(x^k)$ 非奇异时，更新方向 $d^k=-\nabla^2 f(x^k)^{-1}\nabla f(x^k)$. 一般称满足方程(5.4.2)的 d^k 为**牛顿方向**. 因此经典牛顿法的更新格式为

$$x^{k+1}=x^k-\nabla^2 f(x^k)^{-1}\nabla f(x^k). \quad (5.4.3)$$

注意，在格式(5.4.3)中，步长 α_k 恒为 1，即可以不额外考虑步长的选取. 我们也称步长为 1 的牛顿法为经典牛顿法.

5.4.2 收敛性分析

经典牛顿法(5.4.3)有很好的局部收敛性质. 实际上我们有如下定理:

定理 5.6 (经典牛顿法的收敛性) 假设目标函数 f 是二阶连续可微的函数，且海瑟矩阵在最优值点 x^* 的一个邻域 $N_\delta(x^*)$ 内是利普希茨连续的，即存在常数 $L>0$ 使得

$$\|\nabla^2 f(x)-\nabla^2 f(y)\|\leqslant L\|x-y\|,\quad \forall x,y\in N_\delta(x^*).$$

如果函数 $f(x)$ 在点 x^* 处满足 $\nabla f(x^*)=0,\nabla^2 f(x^*)\succ 0$，则对于迭代法(5.4.3)有如下结论:

(1) 如果初始点离 x^* 足够近，则牛顿法产生的迭代点列 $\{x^k\}$ 收敛到 x^*;

(2) $\{x^k\}$ 收敛到 x^* 的速度是 Q-二次的;

(3) $\{\|\nabla f(x^k)\|\}$Q-二次收敛到 *0*.

证明. 从牛顿法的定义(5.4.3)和最优值点 x^* 的性质 $\nabla f(x^*)=0$ 可得

$$\begin{aligned}x^{k+1}-x^* &= x^k-\nabla^2 f(x^k)^{-1}\nabla f(x^k)-x^*\\ &= \nabla^2 f(x^k)^{-1}[\nabla^2 f(x^k)(x^k-x^*)-(\nabla f(x^k)-\nabla f(x^*))].\end{aligned}\tag{5.4.4}$$

根据泰勒公式，可得

$$\nabla f(x^k)-\nabla f(x^*)=\int_0^1\nabla^2 f(x^k+t(x^*-x^k))(x^k-x^*)\mathrm{d}t,$$

因此我们有估计

$$\begin{aligned}&\|\nabla^2 f(x^k)(x^k-x^*)-(\nabla f(x^k)-\nabla f(x^*))\|\\ =&\Big\|\int_0^1[\nabla^2 f(x^k+t(x^*-x^k))-\nabla^2 f(x^k)](x^k-x^*)\mathrm{d}t\Big\|\\ \leqslant&\int_0^1\|\nabla^2 f(x^k+t(x^*-x^k))-\nabla^2 f(x^k)\|\|x^k-x^*\|\mathrm{d}t\\ \leqslant&\|x^k-x^*\|^2\int_0^1 Lt\mathrm{d}t\\ =&\frac{L}{2}\|x^k-x^*\|^2,\end{aligned}\tag{5.4.5}$$

其中第二个不等式是由于海瑟矩阵的局部利普希茨连续性. 又因为 $\nabla^2 f(x^*)$ 是非奇异的且 f 二阶连续可微，因此存在 r，使得对任意满足 $\|x - x^*\| \leqslant r$ 的点 x 均有 $\|\nabla^2 f(x)^{-1}\| \leqslant 2\|\nabla^2 f(x^*)^{-1}\|$. 结合(5.4.4)式与(5.4.5)式可得:

$$\begin{aligned} &\|x^{k+1} - x^*\| \\ \leqslant &\|\nabla^2 f(x^k)^{-1}\| \|\nabla^2 f(x^k)(x^k - x^*) - (\nabla f(x^k) - \nabla f(x^*))\| \\ \leqslant &L\|\nabla^2 f(x^*)^{-1}\| \|x^k - x^*\|^2. \end{aligned} \tag{5.4.6}$$

因此，当初始点 x^0 满足

$$\|x^0 - x^*\| \leqslant \min\left\{\delta, r, \frac{1}{2L\|\nabla^2 f(x^*)^{-1}\|}\right\} \xlongequal{\text{def}} \hat{\delta}$$

时，可保证迭代点列一直处于邻域 $N_{\hat{\delta}}(x^*)$ 中，因此 $\{x^k\}$ Q-二次收敛到 x^*. 由牛顿方程(5.4.2)可知

$$\begin{aligned} \|\nabla f(x^{k+1})\| &= \|\nabla f(x^{k+1}) - \nabla f(x^k) - \nabla^2 f(x^k) d^k\| \\ &= \left\| \int_0^1 \nabla^2 f(x^k + t d^k) d^k dt - \nabla^2 f(x^k) d^k \right\| \\ &\leqslant \int_0^1 \|\nabla^2 f(x^k + t d^k) - \nabla^2 f(x^k)\| \|d^k\| dt \\ &\leqslant \frac{L}{2}\|d^k\|^2 \leqslant \frac{1}{2} L \|\nabla^2 f(x^k)^{-1}\|^2 \|\nabla f(x^k)\|^2 \\ &\leqslant 2L\|\nabla^2 f(x^*)^{-1}\|^2 \|\nabla f(x^k)\|^2. \end{aligned} \tag{5.4.7}$$

这证明了梯度的范数 Q-二次收敛到 0. □

定理5.6表明经典牛顿法是收敛速度很快的算法，但它的收敛是有条件的：第一，初始点 x^0 必须距离问题的解充分近，即牛顿法只有局部收敛性，当 x^0 距问题的解较远时，牛顿算法在多数情况下会失效；第二，海瑟矩阵 $\nabla^2 f(x^*)$ 需要为正定矩阵，有例子表明，若 $\nabla^2 f(x^*)$ 是奇异的半正定矩阵，牛顿算法的收敛速度可能仅达到 Q-线性. 在定理 5.6的证明中还可以看出，问题的条件数并不会在很大程度上影响牛顿法的收敛速度，利普希茨常数 L 在迭代后期通常会被 $\|x^k - x^*\|$ 抵消. 但对于病态问题，牛顿法的收敛域可能会变小，这对初值选取有了更高的要求.

以上总结了牛顿法的特点，我们可以知道牛顿法适用于优化问题的高精度求解，但它没有全局收敛性质. 因此在实际应用中，人们通常会使用梯度类算法先求得较低精度的解，而后调用牛顿法来获得高精度的解.

5.4.3 修正牛顿法

尽管我们提出了算法 (5.4.3) 并分析了其理论性质，在实际应用中此格式几乎是不能使用的. 经典牛顿法有如下缺陷：

(1) 每一步迭代需要求解一个 n 维线性方程组，这导致在高维问题中计算量很大. 海瑟矩阵 $\nabla^2 f(x^k)$ 既不容易计算又不容易储存.

(2) 当 $\nabla^2 f(x^k)$ 不正定时，由牛顿方程 (5.4.2) 给出的解 d^k 的性质通常比较差. 例如可以验证当海瑟矩阵正定时，d^k 是一个下降方向，而在其他情况下 d^k 不一定为下降方向.

(3) 当迭代点距最优值较远时，直接选取步长 $\alpha_k = 1$ 会使得迭代极其不稳定，在有些情况下迭代点列会发散.

为了克服这些缺陷，我们必须对经典牛顿法做出某种修正或变形，使其成为真正可以使用的算法. 这里介绍带线搜索的修正牛顿法，其基本思想是对牛顿方程中的海瑟矩阵 $\nabla^2 f(x^k)$ 进行修正，使其变成正定矩阵；同时引入线搜索以改善算法稳定性. 它的一般框架见算法 5.3. 该算法的关键在于修

算法 5.3 带线搜索的修正牛顿法

1. 给定初始点 x^0.
2. **for** $k = 0, 1, 2, \cdots$ **do**
3. 确定矩阵 E^k 使得矩阵 $B^k \overset{\text{def}}{=\!=} \nabla^2 f(x^k) + E^k$ 正定且条件数较小.
4. 求解修正的牛顿方程 $B^k d^k = -\nabla f(x^k)$ 得方向 d^k.
5. 使用任意一种线搜索准则确定步长 α_k.
6. 更新 $x^{k+1} = x^k + \alpha_k d^k$.
7. **end for**

正矩阵 E^k 如何选取. 一个最直接的取法是取 $E^k = \tau_k I$，即取 E^k 为单位矩阵的常数倍. 根据矩阵理论可以知道，当 τ_k 充分大时，总可以保证 B^k 是正定矩阵. 然而 τ_k 不宜取得过大，这是因为当 τ_k 趋于无穷时，d^k 的方向会接近负梯度方向. 比较合适的取法是先估计 $\nabla^2 f(x^k)$ 的最小特征值，再适当选择 τ_k. 除此之外，E^k 可以采用修正 Cholesky 分解进行隐式更新，读者可参考 [52].

5.4.4 非精确牛顿法

在经典牛顿法中，计算牛顿方向 d^k 依赖于求解线性方程组. 当 n 较大但 $\nabla^2 f(x^k)$ 有稀疏结构时，需要通过迭代法来求解牛顿方程(5.4.2). 我们知道迭代法求解线性方程组总有精度误差，那么牛顿方程解的误差对牛顿法收敛性有何影响? 如何控制解的精度使得牛顿法依然能够收敛? 下面将简要回答这一问题.

考虑牛顿方程(5.4.2)的非精确解 d^k，我们引入向量 r^k 来表示残差，则非精确牛顿方向满足

$$\nabla^2 f(x^k)d^k = -\nabla f(x^k) + r^k. \tag{5.4.8}$$

这里假设相对误差 η_k 满足

$$\|r^k\| \leqslant \eta_k \|\nabla f(x^k)\|. \tag{5.4.9}$$

显然，牛顿法的收敛性依赖于相对误差 η_k 的选取，直观上牛顿方程求解得越精确，非精确牛顿法的收敛性就越好. 为此我们直接给出如下的定理（证明见 [120]）:

定理 5.7 (非精确牛顿法的收敛性) *设函数 $f(x)$ 二阶连续可微，且在最小值点 x^* 处的海瑟矩阵正定，则在非精确牛顿法中，*

(1) 若存在常数 $t < 1$ 使得 η_k 满足 $0 < \eta_k < t, k = 1, 2, \cdots$，则该算法收敛速度是 Q-线性的;

(2) 若 $\lim\limits_{k\to\infty} \eta_k = 0$，则该算法收敛速度是 Q-超线性的;

(3) 若 $\eta_k = \mathcal{O}(\|\nabla f(x^k)\|)$，则该算法收敛速度是 Q-二次的.

定理 5.7的直观含义是如果要达到更好的收敛性就必须使得牛顿方程求解更加精确. 在一般迭代法中，算法的停机准则通常都会依赖于相对误差的大小. 定理 5.7 的第一条表明我们完全可以将这个相对误差设置为固定值，算法依然有收敛性. 和经典牛顿法相比，固定误差的非精确牛顿法仅仅有 Q-线性收敛性，但在病态问题上的表现很可能好于传统的梯度法. 如果希望非精确牛顿法能有 Q-二次收敛速度，则在迭代后期牛顿方程必须求解足够精确，这在本质上和牛顿法并无差别.

常用的非精确牛顿法是牛顿共轭梯度法，即使用共轭梯度法求解 (5.4.2) 式. 由于共轭梯度法在求解线性方程组方面有不错的表现，因此只需少数几

步（有时可能只需要一步）迭代就可以达到定理 5.7中第一条结论需要的条件. 在多数问题上牛顿共轭梯度法都有较好的数值表现，该方法已经是求解优化问题不可少的优化工具.

5.4.5 应用举例

本小节给出牛顿法的一个具体例子，从而说明牛顿法具体的迭代过程. 考虑二分类的逻辑回归模型：

$$\min_{x} \quad \ell(x) \xlongequal{\text{def}} \frac{1}{m}\sum_{i=1}^{m}\ln(1+\exp(-b_i a_i^{\mathrm{T}}x)) + \lambda\|x\|_2^2. \tag{5.4.10}$$

在实际中，λ 经常取为 $\dfrac{1}{100m}$. 接下来推导牛顿法，这又化为了计算目标函数 $\ell(x)$ 的梯度和海瑟矩阵的问题. 根据向量值函数求导法，容易算出

$$\begin{aligned}\nabla\ell(x) &= \frac{1}{m}\sum_{i=1}^{m}\frac{1}{1+\exp(-b_i a_i^{\mathrm{T}}x)}\cdot\exp(-b_i a_i^{\mathrm{T}}x)\cdot(-b_i a_i) + 2\lambda x \\ &= -\frac{1}{m}\sum_{i=1}^{m}(1-p_i(x))b_i a_i + 2\lambda x,\end{aligned} \tag{5.4.11}$$

其中 $p_i(x) = \dfrac{1}{1+\exp(-b_i a_i^{\mathrm{T}}x)}$. 再对(5.4.11)式求导可得到

$$\begin{aligned}\nabla^2\ell(x) &= \frac{1}{m}\sum_{i=1}^{m}b_i\cdot\nabla p_i(x)a_i^{\mathrm{T}} + 2\lambda I \\ &= \frac{1}{m}\sum_{i=1}^{m}b_i\frac{-1}{(1+\exp(-b_i a_i^{\mathrm{T}}x))^2}\cdot\exp(-b_i a_i^{\mathrm{T}}x)\cdot(-b_i a_i a_i^{\mathrm{T}}) + 2\lambda I \\ &= \frac{1}{m}\sum_{i=1}^{m}(1-p_i(x))p_i(x)a_i a_i^{\mathrm{T}} + 2\lambda I,\end{aligned} \tag{5.4.12}$$

其中利用了 $b_i^2 = 1$ 以及

$$\frac{\exp(-b_i a_i^{\mathrm{T}}x)}{(1+\exp(-b_i a_i^{\mathrm{T}}x))^2} = p_i(x)(1-p_i(x)).$$

实际上我们可以使用矩阵语言将以上结果用更紧凑的形式表达. 引入矩阵 $A = [a_1, a_2, \cdots, a_m]^{\mathrm{T}} \in \mathbb{R}^{m\times n}$，向量 $b = (b_1, b_2, \cdots, b_m)^{\mathrm{T}}$，以及

$$p(x) = (p_1(x), p_2(x), \cdots, p_m(x))^{\mathrm{T}},$$

梯度和海瑟矩阵可重写为

$$\begin{aligned}\nabla \ell(x) &= -\frac{1}{m} A^{\mathrm{T}}(b - b \odot p(x)) + 2\lambda x, \\ \nabla^2 \ell(x) &= \frac{1}{m} A^{\mathrm{T}} W(x) A + 2\lambda I,\end{aligned} \tag{5.4.13}$$

其中 $W(x)$ 为由 $\{p_i(x)(1-p_i(x))\}_{i=1}^m$ 生成的对角矩阵，$\odot$ 表示两个向量逐分量的乘积. 因此牛顿法可以写作

$$x^{k+1} = x^k + \left(\frac{1}{m}(A^{\mathrm{T}} W(x^k) A + 2\lambda I\right)^{-1} \left(\frac{1}{m} A^{\mathrm{T}}(b - b \odot p(x^k)) - 2\lambda x^k\right).$$

当变量规模不是很大时，可以利用正定矩阵的 Cholesky 分解来求解牛顿方程；当变量规模较大时，可以使用共轭梯度法进行不精确求解.

这里采用 LIBSVM[29] 网站的数据集，见表5.1. 对于不同的数据集均调

表 5.1 数据集信息

名称	m	n
a9a	16281	122
ijcnn1	91701	22
CINA	3206	132

用牛顿法求解，其迭代过程见图 5.10. 其中，我们采用不精确的共轭梯度法求解牛顿方程，使得如下条件成立：

$$\|\nabla^2 \ell(x^k) d^k + \nabla \ell(x^k)\|_2 \leqslant \min\left\{\|\nabla \ell(x^k)\|_2^2, 0.1\|\nabla \ell(x^k)\|_2\right\}.$$

从图5.10中可以看到，在精确解附近梯度范数具有 Q-超线性收敛性.

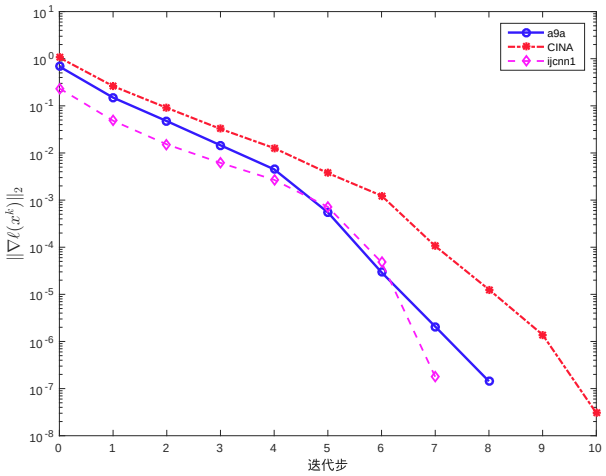


图 5.10 梯度范数随迭代步的变化

5.5 拟牛顿类算法

牛顿法在理论上和实践中均取得很好的效果. 然而对于大规模问题，函数的海瑟矩阵计算代价特别大或者难以得到，即便得到海瑟矩阵我们还需要求解一个大规模线性方程组. 那么能否使用海瑟矩阵或其逆矩阵的近似来进行牛顿迭代呢? 拟牛顿法便是这样的算法，它能够在每一步以较小的计算代价生成近似矩阵，并且使用近似矩阵代替海瑟矩阵而产生的迭代序列仍具有超线性收敛的性质.

拟牛顿方法不计算海瑟矩阵 $\nabla^2 f(x)$，而是构造其近似矩阵 B^k 或其逆的近似矩阵 H^k. 我们希望 B^k 或 H^k 仍然保留海瑟矩阵的部分性质，例如使得 d^k 仍然为下降方向. 那么拟牛顿矩阵应该满足一些什么性质? 如何构造它们呢?

5.5.1 割线方程

首先回顾牛顿法的推导过程. 设 $f(x)$ 是二次连续可微函数，根据泰勒展开，向量值函数 $\nabla f(x)$ 在点 x^{k+1} 处的近似为

$$\nabla f(x) = \nabla f(x^{k+1}) + \nabla^2 f(x^{k+1})(x - x^{k+1}) + \mathcal{O}(\|x - x^{k+1}\|^2). \tag{5.5.1}$$

令 $x = x^k, s^k = x^{k+1} - x^k$ 及 $y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$，得到

$$\nabla^2 f(x^{k+1}) s^k + \mathcal{O}(\|s^k\|^2) = y^k. \tag{5.5.2}$$

忽略高阶项 $\|s^k\|^2$，我们希望海瑟矩阵的近似矩阵 B^{k+1} 满足方程

$$y^k = B^{k+1} s^k, \tag{5.5.3}$$

或者其逆的近似矩阵 H^{k+1} 满足方程

$$s^k = H^{k+1} y^k, \tag{5.5.4}$$

并称(5.5.3)式和(5.5.4)式为**割线方程**.

还可以从另一个角度理解割线方程(5.5.3). 我们知道，牛顿法本质上是对目标函数 $f(x)$ 在迭代点 x^k 处做二阶近似然后求解. 考虑在点 x^{k+1} 处的二阶近似

$$m_{k+1}(d) = f(x^{k+1}) + \nabla f(x^{k+1})^{\mathrm{T}} d + \frac{1}{2} d^{\mathrm{T}} B^{k+1} d, \tag{5.5.5}$$

我们要求 $m_{k+1}(d)$ 在 $d = -s^k$ 和 $d = 0$ 处的梯度与 $f(x)$ 在 $x = x^k$ 和 $x = x^{k+1}$ 处的梯度分别保持一致. 注意到 $\nabla m_{k+1}(0) = \nabla f(x^{k+1})$ 是自然满足的，为了使得第一个条件满足只需

$$\nabla m_{k+1}(-s^k) = \nabla f(x^{k+1}) - B^{k+1} s^k = \nabla f(x^k), \tag{5.5.6}$$

整理即可得到(5.5.3)式.

另外，注意到近似矩阵 B^k 的正定性是一个很关键的因素，在(5.5.3)式两边同时左乘 $(s^k)^{\mathrm{T}}$ 可得 $(s^k)^{\mathrm{T}} B^{k+1} s^k = (s^k)^{\mathrm{T}} y^k$，因此条件

$$(s^k)^{\mathrm{T}} y^k > 0 \tag{5.5.7}$$

为 B^{k+1} 正定的一个必要条件. 我们额外要求条件(5.5.7)在迭代过程中始终满足，这个条件也称为**曲率条件**. 对于一般的目标函数 $f(x)$，需要使用 Wolfe 准则线搜索来保证曲率条件(5.5.7)成立. 实际上，根据 Wolfe 准则的条件(5.1.4b)有 $\nabla f(x^{k+1})^{\mathrm{T}} s^k \geqslant c_2 \nabla f(x^k)^{\mathrm{T}} s^k$，两边同时减去 $\nabla f(x^k)^{\mathrm{T}} s^k$，

$$(y^k)^{\mathrm{T}} s^k \geqslant (c_2 - 1) \nabla f(x^k)^{\mathrm{T}} s^k > 0,$$

这是因为 $c_2 < 1$ 以及 $s^k = \alpha_k d^k$ 是下降方向. 仅仅使用 Armijo 准则不能保证曲率条件成立.

在通常情况下，近似矩阵 B^{k+1} 或 H^{k+1} 是由上一步迭代加上一个修正得到的，并且要求满足割线方程(5.5.3)．这一小节先给出拟牛顿方法的一般框架（算法5.4），下一小节将讨论一些具体的矩阵更新方式．

算法 5.4 拟牛顿算法框架

1. 给定 $x^0 \in \mathbb{R}^n$, 初始矩阵 $B^0 \in \mathbb{R}^{n\times n}$（或 H^0），令 $k = 0$.
2. **while** 未达到停机准则 **do**
3. 计算方向 $d^k = -(B^k)^{-1}\nabla f(x^k)$ 或 $d^k = -H^k\nabla f(x^k)$.
4. 通过线搜索找到合适的步长 $\alpha_k > 0$，令 $x^{k+1} = x^k + \alpha_k d^k$.
5. 更新海瑟矩阵的近似矩阵 B^{k+1} 或其逆矩阵的近似 H^{k+1}.
6. $k \leftarrow k + 1$.
7. **end while**

在实际应用中基于 H^k 的拟牛顿法更加实用，这是因为根据 H^k 计算下降方向 d^k 不需要求解线性方程组，而求解线性方程组在大规模问题上是非常耗时的．但基于 B^k 的拟牛顿法有比较好的理论性质，产生的迭代序列比较稳定．如果有办法快速求解线性方程组，我们也可采用基于 B^k 的拟牛顿法．此外在某些场景下，比如有些带约束的优化问题的算法设计，由于需要用到海瑟矩阵的近似，B^k 的使用也很常见．

5.5.2 拟牛顿矩阵更新方式

这一小节介绍一些常见的拟牛顿矩阵的更新方式．

1. 秩一更新 (SR1)

秩一更新 (SR1) 公式是结构最简单的拟牛顿矩阵更新公式．设 B^k 是第 k 步的近似海瑟矩阵，我们通过对 B^k 进行秩一修正得到 B^{k+1}，使其满足割线方程(5.5.3)．为此使用待定系数法求出修正矩阵，并设

$$B^{k+1} = B^k + auu^{\mathrm{T}}, \tag{5.5.8}$$

其中 $u \in \mathbb{R}^n, a \in \mathbb{R}$ 待定．根据割线方程(5.5.3)，

$$B^{k+1}s^k = (B^k + auu^{\mathrm{T}})s^k = y^k,$$

进而得到

$$(a \cdot u^{\mathrm{T}}s^k)u = y^k - B^k s^k.$$

注意到 $a \cdot u^{\mathrm{T}} s^k$ 是一个标量，因此 u 和 $y^k - B^k s^k$ 方向相同. 不妨令 $u = y^k - B^k s^k$，代入原方程可知

$$a((y^k - B^k s^k)^{\mathrm{T}} s^k)(y^k - B^k s^k) = y^k - B^k s^k.$$

如果假设 $(y^k - B^k s^k)^{\mathrm{T}} s^k \neq 0$，可以得到 $a = \dfrac{1}{(y^k - B^k s^k)^{\mathrm{T}} s^k}$，最终得到更新公式为

$$B^{k+1} = B^k + \frac{(y^k - B^k s^k)(y^k - B^k s^k)^{\mathrm{T}}}{(y^k - B^k s^k)^{\mathrm{T}} s^k}. \tag{5.5.9}$$

我们称(5.5.9)式为基于 B^k 的 SR1 公式. 由完全一样的过程我们可以根据割线方程(5.5.4)得到基于 H^k 的 SR1 公式：

$$H^{k+1} = H^k + \frac{(s^k - H^k y^k)(s^k - H^k y^k)^{\mathrm{T}}}{(s^k - H^k y^k)^{\mathrm{T}} y^k}. \tag{5.5.10}$$

SR1 公式虽然结构简单，但是有一个重大缺陷：它不能保证矩阵在迭代过程中保持正定. 容易验证 $(y^k - B^k s^k)^{\mathrm{T}} s^k > 0$ 是 B^{k+1} 正定的一个充分条件，但这个条件在迭代过程中未必得到满足. 因此在实际中较少使用 SR1 公式.

另一个比较有趣的观察是公式(5.5.9)和(5.5.10)在形式上互为对偶. 将公式(5.5.9)里的变量作如下替换：

$$B^k \to H^k, \quad s^k \leftrightarrow y^k,$$

即可得到公式(5.5.10).

2. BFGS 公式

为了克服 SR1 公式的缺陷，现在考虑对 B^k 的秩二更新. 同样地，采用待定系数法来推导此公式. 设

$$B^{k+1} = B^k + auu^{\mathrm{T}} + bvv^{\mathrm{T}}, \tag{5.5.11}$$

其中 $u, v \in \mathbb{R}^n$，$a, b \in \mathbb{R}$ 待定. 根据割线方程(5.5.3)，

$$B^{k+1} s^k = (B^k + auu^{\mathrm{T}} + bvv^{\mathrm{T}}) s^k = y^k,$$

整理可得

$$(a \cdot u^{\mathrm{T}} s^k) u + (b \cdot v^{\mathrm{T}} s^k) v = y^k - B^k s^k.$$

我们通过选取 u 和 v 让以上等式成立即可. 实际上, u,v 有非常多的取法, 一种最直接的取法是让上面等式左右两边的两项分别对应相等, 即

$$u = y^k, \quad a \cdot u^{\mathrm{T}} s^k = 1,$$
$$v = B^k s^k, \quad b \cdot v^{\mathrm{T}} s^k = -1.$$

因此得到更新方式

$$B^{k+1} = B^k + \frac{y^k (y^k)^{\mathrm{T}}}{(s^k)^{\mathrm{T}} y^k} - \frac{B^k s^k (B^k s^k)^{\mathrm{T}}}{(s^k)^{\mathrm{T}} B^k s^k}. \tag{5.5.12}$$

格式(5.5.12)被称为基于 B^k 的 BFGS 公式, 它是由 Broyden, Fletcher, Goldfarb, Shanno 四人名字的首字母组成.

为了推导基于 H^k 的 BFGS 公式, 我们需要如下的 Sherman-Morrison-Woodbury（简称 SMW）公式来给出原矩阵和其在秩-k 更新后的矩阵的逆矩阵之间的关系.

命题 5.1 (SMW 公式) 设 $A \in \mathbb{R}^{n\times n}$ 为一可逆矩阵, 给定矩阵 $U \in \mathbb{R}^{n\times k}$, $C \in \mathbb{R}^{k\times k}$, $V \in \mathbb{R}^{k\times n}$ 且 C 可逆. 那么 $A + UCV$ 可逆当且仅当 $C^{-1} + VA^{-1}U$ 可逆, 且此时 $A + UCV$ 的逆矩阵可以表示为

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}. \tag{5.5.13}$$

根据 SMW 公式并假设 $H^k = (B^k)^{-1}$, 可立即推出基于 H^k 的 BFGS 公式:

$$H^{k+1} = (I - \rho_k y^k (s^k)^{\mathrm{T}})^{\mathrm{T}} H^k (I - \rho_k y^k (s^k)^{\mathrm{T}}) + \rho_k s^k (s^k)^{\mathrm{T}}, \tag{5.5.14}$$

其中 $\rho_k = \dfrac{1}{(s^k)^{\mathrm{T}} y^k}$. 容易看出, 若要 BFGS 公式更新产生的矩阵 H^{k+1} 正定, 一个充分条件是不等式(5.5.7)成立且上一步更新矩阵 H^k 正定. 在问题求解过程中, 条件(5.5.7) 不一定会得到满足, 此时应该使用 Wolfe 准则的线搜索来迫使条件(5.5.7)成立.

BFGS 公式是目前最有效的拟牛顿更新格式之一, 它有比较好的理论性质, 实现起来也并不复杂. 对格式(5.5.14)进行改动可得到有限内存 BFGS 格式 (L-BFGS), 它是常用的处理大规模优化问题的拟牛顿类算法, 我们将在本节的末尾介绍这一算法.

3. DFP 公式

在 BFGS 公式的推导中，如果利用割线方程(5.5.4)对 H^k 推导秩二修正的拟牛顿修正，我们将得到基于 H^k 的拟牛顿矩阵更新

$$H^{k+1} = H^k - \frac{H^k y^k (H^k y^k)^{\mathrm{T}}}{(y^k)^{\mathrm{T}} H^k y^k} + \frac{s^k (s^k)^{\mathrm{T}}}{(y^k)^{\mathrm{T}} s^k}. \tag{5.5.15}$$

这种迭代格式首先由 Davidon 发现，此后由 Fletcher 以及 Powell 进一步发展，因此被称为 DFP 公式. 根据 SMW 公式可得其关于 B^k 的更新格式

$$B^{k+1} = (I - \rho_k s^k (y^k)^{\mathrm{T}})^{\mathrm{T}} B^k (I - \rho_k s^k (y^k)^{\mathrm{T}}) + \rho_k y^k (y^k)^{\mathrm{T}}, \tag{5.5.16}$$

其中 ρ_k 的定义同(5.5.14)式.

可以看到，DFP 公式(5.5.15)(5.5.16)和 BFGS 公式(5.5.12)(5.5.14) 分别呈对偶关系. 将 BFGS 格式(5.5.14)中的 H^k 换成 B^k，s^k 与 y^k 对换便得到了 DFP 格式(5.5.16).

遗憾的是，尽管 DFP 格式在很多方面和 BFGS 格式存在对偶关系，但从实际效果来看，DFP 格式整体上不如 BFGS 格式. 因此在实际使用中人们更多使用 BFGS 格式.

5.5.3 拟牛顿法的全局收敛性

本小节介绍拟牛顿法的收敛性质. 首先我们利用 Zoutendijk 条件得到拟牛顿法基本的收敛性，之后简要介绍收敛速度.

定理 5.8 (BFGS 全局收敛性 [98]定理 6.5) *假设初始矩阵 B^0 是对称正定矩阵，目标函数 $f(x)$ 是二阶连续可微函数，且下水平集*

$$\mathcal{L} = \{x \in \mathbb{R}^n \mid f(x) \leqslant f(x^0)\}$$

是凸的，并且存在正数 m 以及 M 使得对于任意的 $z \in \mathbb{R}^n$ 以及任意的 $x \in \mathcal{L}$ 有

$$m\|z\|^2 \leqslant z^{\mathrm{T}} \nabla^2 f(x) z \leqslant M\|z\|^2, \tag{5.5.17}$$

则采用 BFGS 格式(5.5.12) 并结合 Wolfe 线搜索的拟牛顿算法全局收敛到 $f(x)$ 的极小值点 x^*.

定理5.8叙述了 BFGS 格式的全局收敛性，但没有说明以什么速度收敛. 下面这个定理介绍了在一定条件下 BFGS 格式会达到 Q-超线性收敛速度. 这里只给出定理结果，详细的证明过程可参考 [98].

定理 5.9 (BFGS 收敛速度) 设 $f(x)$ 二阶连续可微，在最优点 x^* 的一个邻域内海瑟矩阵利普希茨连续，且使用 BFGS 迭代格式收敛到 f 的最优值点 x^*. 若迭代点列 $\{x^k\}$ 满足

$$\sum_{k=1}^{\infty} \|x^k - x^*\| < +\infty, \tag{5.5.18}$$

则 $\{x^k\}$ 以 Q-超线性收敛到 x^*.

正如我们预期的，由于仅仅使用了海瑟矩阵的近似矩阵，拟牛顿法只能达到 Q-超线性收敛速度，这个速度和牛顿法相比较慢. 但由于拟牛顿法不需要每一步都计算海瑟矩阵，它在整体计算开销方面可能远远小于牛顿法，因此在实际问题中较为实用. 同样地，定理5.9的结果建立在序列 $\{x^k\}$ 本身收敛的前提之上，而对于强凸函数，定理5.8 保证了序列 $\{x^k\}$ 是全局收敛的. 如果函数的性质稍差，则拟牛顿法可能只有局部的收敛性.

5.5.4 有限内存 BFGS 方法

拟牛顿法虽然克服了计算海瑟矩阵的困难，但是它仍然无法应用在大规模优化问题上. 一般来说，拟牛顿矩阵 B^k 或 H^k 是稠密矩阵，而存储稠密矩阵要消耗 $\mathcal{O}(n^2)$ 的内存，这对于大规模问题显然是不可能实现的. 在本小节介绍的有限内存 BFGS 方法（L-BFGS）解决了这一存储问题，从而使得人们在大规模问题上也可应用拟牛顿类方法加速迭代的收敛.

L-BFGS 方法是根据 BFGS 公式(5.5.12)(5.5.14)变形而来的. 为了推导方便，我们以 H^k 的更新公式(5.5.14)为基础来推导相应的 L-BFGS 公式. 首先引入新的记号重写(5.5.14)式：

$$H^{k+1} = (V^k)^{\mathrm{T}} H^k V^k + \rho_k s^k (s^k)^{\mathrm{T}}, \tag{5.5.19}$$

其中

$$\rho_k = \frac{1}{(y^k)^{\mathrm{T}} s^k}, \quad V^k = I - \rho_k y^k (s^k)^{\mathrm{T}}. \tag{5.5.20}$$

观察到(5.5.19)式有类似递推的性质，为此我们可将(5.5.19)式递归地展开 m

次，其中 m 是一个给定的整数：

$$\begin{aligned} & H^k \\ =&(V^{k-m}\cdots V^{k-1})^{\mathrm{T}}H^{k-m}(V^{k-m}\cdots V^{k-1})+ \\ & \rho_{k-m}(V^{k-m+1}\cdots V^{k-1})^{\mathrm{T}}s^{k-m}(s^{k-m})^{\mathrm{T}}(V^{k-m+1}\cdots V^{k-1})+ \\ & \rho_{k-m+1}(V^{k-m+2}\cdots V^{k-1})^{\mathrm{T}}s^{k-m+1}(s^{k-m+1})^{\mathrm{T}}(V^{k-m+2}\cdots V^{k-1})+ \\ & \cdots+\rho_{k-1}s^{k-1}(s^{k-1})^{\mathrm{T}}. \end{aligned} \tag{5.5.21}$$

为了达到节省内存的目的，(5.5.19)式不能无限展开下去，但这会产生一个问题：H^{k-m} 还是无法显式求出. 一个很自然的想法就是用 H^{k-m} 的近似矩阵来代替 H^{k-m} 进行计算，近似矩阵的选取方式非常多，但基本原则是要保证近似矩阵具有非常简单的结构. 假定我们给出了 H^{k-m} 的一个近似矩阵 $\hat{H}^{k-m}$，(5.5.21)式便可以用于计算拟牛顿迭代.

在拟牛顿迭代中，实际上并不需要计算 H^k 的显式形式，只需要利用 $H^k\nabla f(x^k)$ 来计算迭代方向 d^k. 为此先直接给出一个利用展开式(5.5.21)直接求解 $H^k\nabla f(x^k)$ 的算法，见算法 5.5. 该算法的设计非常巧妙，它充分利用

算法 5.5 L-BFGS 双循环递归算法

1. 初始化 $q \leftarrow \nabla f(x^k)$.
2. **for** $i = k-1, k-2, \cdots, k-m$ **do**
3. 计算并保存 $\alpha_i \leftarrow \rho_i (s^i)^{\mathrm{T}} q$.
4. 更新 $q \leftarrow q - \alpha_i y^i$.
5. **end for**
6. 初始化 $r \leftarrow \hat{H}^{k-m} q$，其中 $\hat{H}^{k-m}$ 是 H^{k-m} 的近似矩阵.
7. **for** $i = k-m, k-m+1, \cdots, k-1$ **do**
8. 计算 $\beta \leftarrow \rho_i (y^i)^{\mathrm{T}} r$.
9. 更新 $r \leftarrow r + (\alpha_i - \beta) s^i$.
10. **end for**
11. 输出 r，即 $H^k \nabla f(x^k)$.

了(5.5.21)式的结构来尽量节省计算 $H^k\nabla f(x^k)$ 的开销. 由于其主体结构包含了方向相反的两个循环，因此它也被称为双循环递归算法.

我们现在给出算法5.5的一个比较直观的执行过程. 在(5.5.21)式中，等式左右两边同时右乘 $\nabla f(x^k)$，若只观察等式右侧，则需要计算

$$V^{k-1}\nabla f(x^k),\ V^{k-2}V^{k-1}\nabla f(x^k),\ \cdots,\ V^{k-m}\cdots V^{k-2}V^{k-1}\nabla f(x^k).$$

这些结果可以递推地进行，无需重复计算. 另一个比较重要的观察是，在计算 $V^{k-l}\cdots V^{k-1}\nabla f(x^k)$ 的过程中恰好同时计算了上一步的 $\rho_{k-l}(s^{k-l})^{\mathrm{T}}[V^{k-l+1}\cdots V^{k-1}\nabla f(x^k)]$，这是一个标量，对应着算法5.5的 α_i. 因此执行完第一个循环后，我们得到了 α_i, q，公式(5.5.21) 变成了如下形式：

$$\begin{aligned} H^k\nabla f(x^k) =&(V^{k-m}\cdots V^{k-1})^{\mathrm{T}}H^{k-m}q+\\ &(V^{k-m+1}\cdots V^{k-1})^{\mathrm{T}}s^{k-m}\alpha_{k-m}+\\ &(V^{k-m+2}\cdots V^{k-1})^{\mathrm{T}}s^{k-m+1}\alpha_{k-m+1}+\cdots+s^{k-1}\alpha_{k-1}. \end{aligned} \tag{5.5.22}$$

公式(5.5.22)已经简化了不少，接下来算法5.5的第二个循环就是自上而下合并每一项. 以合并前两项为例，它们有公共的因子 $(V^{k-m+1}\cdots V^{k-1})^{\mathrm{T}}$，提取出来之后前两项的和可以写为

$$\begin{aligned} &(V^{k-m+1}\cdots V^{k-1})^{\mathrm{T}}((V^{k-m})^{\mathrm{T}}r+\alpha_{k-m}s^{k-m})\\ =&(V^{k-m+1}\cdots V^{k-1})^{\mathrm{T}}(r+(\alpha_{k-m}-\beta)s^{k-m}), \end{aligned}$$

这正是第二个循环的迭代格式. 注意合并后(5.5.22)式的结构仍不变，因此可递归地计算下去，最后变量 r 就是我们期望的结果 $H^k\nabla f(x^k)$.

算法5.5双循环约需要 $4mn$ 次乘法运算与 $2mn$ 次加法运算，若近似矩阵 $\hat{H}^{k-m}$ 是对角矩阵，则额外需要 n 次乘法运算. 由于 m 不会很大，因此该算法的复杂度是 $\mathcal{O}(mn)$. 算法所需要的额外存储为临时变量 α_i，它的大小是 $\mathcal{O}(m)$. 综上所述，L-BFGS 双循环算法是非常高效的.

近似矩阵 $\hat{H}^{k-m}$ 的取法可以是对角矩阵 $\hat{H}^{k-m}=\gamma_k I$，其中

$$\gamma_k=\frac{(s^{k-1})^{\mathrm{T}}y^{k-1}}{(y^{k-1})^{\mathrm{T}}y^{k-1}}. \tag{5.5.23}$$

注意，这恰好是 BB 方法的第一个步长，见(5.2.9)式.

正因为 L-BFGS 方法的出现，人们可以使用拟牛顿类算法求解优化问题. 虽然有关 L-BFGS 方法的收敛性质依然很有限，但在实际应用中 L-BFGS 方法很快成为了应用最广泛的拟牛顿类算法. 比较有趣的是，尽管 DFP 公式和 BFGS 公式呈对偶关系，但极少有人研究有限内存的 DFP 格式，这也使得 BFGS 格式在地位上比 DFP 格式略胜一筹.

算法 5.6 L-BFGS 方法

1. 选择初始点 x^0，参数 $m > 0$，$k \leftarrow 0$.
2. **while** 未达到收敛准则 **do**
3. 选取近似矩阵 $\hat{H}^{k-m}$.
4. 使用算法5.5计算下降方向 $d^k = -H^k \nabla f(x^k)$.
5. 使用线搜索算法计算满足 Wolfe 准则的步长 α_k.
6. 更新 $x^{k+1} = x^k + \alpha_k d^k$.
7. **if** $k > m$ **then**
8. 从内存空间中删除 s^{k-m}, y^{k-m}.
9. **end if**
10. 计算并保存 $s^k = x^{k+1} - x^k$，$y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$.
11. $k \leftarrow k + 1$.
12. **end while**

5.5.5 应用举例

1. 逻辑回归问题

考虑逻辑回归问题

$$\min_x \quad \ell(x) \xlongequal{\text{def}} \frac{1}{m}\sum_{i=1}^{m} \ln(1 + \exp(-b_i a_i^{\mathrm{T}} x)) + \lambda \|x\|_2^2, \tag{5.5.24}$$

这里，选取 $\lambda = \dfrac{1}{100m}$. 目标函数的导数计算可以参考上一节. 同样地，我们选取 LIBSVM 上的数据集，调用 L-BFGS（内存长度取为 5）求解代入数据集后的问题(5.5.24)，其迭代收敛过程见图 5.11.

从 ijcnn 数据集的迭代结果可以看到，当靠近最优解时，L-BFGS 方法的迭代点列呈 Q-线性收敛. 对于 a9a 和 CINA 数据集，由于对应的海瑟矩阵的谱分布不同，图中的迭代还没有进入 LBFGS 算法的线性收敛区域，因而会产生一些抖动，但总体是呈下降趋势.

5.6 信赖域算法

本节介绍信赖域算法. 它和线搜索算法都是借助泰勒展开来对目标函数进行局部近似，但它们看待近似函数的方式不同. 在线搜索算法中，我们先

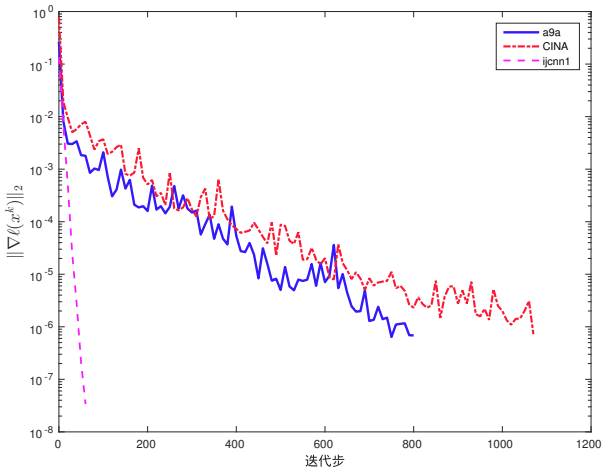


图 5.11 逻辑回归问题

利用近似模型求出下降方向，然后给定步长；而在信赖域类算法中，我们直接在一个有界区域内求解这个近似模型，而后迭代到下一个点. 因此信赖域算法实际上是同时选择了方向和步长.

5.6.1 信赖域算法框架

我们对信赖域算法给一个直观的数学表达. 根据带拉格朗日余项的泰勒展开，

$$f(x^k+d)=f(x^k)+\nabla f(x^k)^{\mathrm{T}}d+\frac{1}{2}d^{\mathrm{T}}\nabla^2 f(x^k+td)d,$$

其中 $t\in(0,1)$ 为和 d 有关的正数. 和牛顿法相同，我们利用 $f(x)$ 的一个二阶近似来刻画 $f(x)$ 在点 $x=x^k$ 处的性质：

$$m_k(d)=f(x^k)+\nabla f(x^k)^{\mathrm{T}}d+\frac{1}{2}d^{\mathrm{T}}B^k d, \tag{5.6.1}$$

其中 B^k 是对称矩阵. 这里要求 B^k 是海瑟矩阵的近似矩阵，如果 B^k 恰好是函数 $f(x)$ 在点 $x=x^k$ 处的海瑟矩阵，那么当 $f(x)$ 充分光滑时，$m_k(d)$ 的逼近误差是 $O(\|d\|^3)$.

我们使用了二阶泰勒展开来近似目标函数 $f(x)$，但还需要考虑到泰勒展开是函数的局部性质，它仅仅对模长较小的 d 有意义. 当 d 过长时，模型(5.6.1) 便不再能刻画 $f(x)$ 的特征，为此需要对模型(5.6.1)添加约束. 我们

仅在如下球内考虑 $f(x)$ 的近似：

$$\Omega_k = \{x^k + d \mid \|d\| \leqslant \Delta_k\},$$

其中 $\Delta_k > 0$ 是一个和迭代有关的参数. 我们称 Ω_k 为**信赖域**，Δ_k 为 **信赖域半径**. 从命名方式也可看出，信赖域就是我们相信 $m_k(d)$ 能很好地近似 $f(x)$ 的区域，而 Δ_k 表示了这个区域的大小. 因此信赖域算法每一步都需要求解如下子问题

$$\min_{d\in\mathbb{R}^n} \quad m_k(d), \quad \text{s.t.} \quad \|d\| \leqslant \Delta_k. \tag{5.6.2}$$

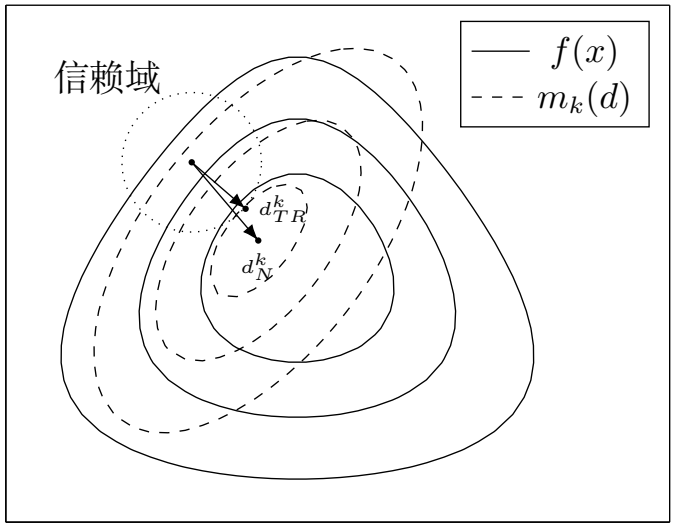


图 5.12 信赖域算法一步迭代

图5.12显示了子问题(5.6.2)的求解过程. 图中实线表示 $f(x)$ 的等高线，虚线表示 $m_k(d)$ 的等高线（这里有关系 $d = x - x^k$），d_N^k 表示求解无约束问题得到的下降方向（若 B^k 为海瑟矩阵则 d_N^k 为牛顿方向），d_{TR}^k 表示求解信赖域子问题(5.6.2)得到的下降方向，可以看到二者明显是不同的. 信赖域算法正是限制了 d 的大小，使得迭代更加保守，因此可以在牛顿方向很差时发挥作用.

在子问题(5.6.2)中仍需要确定信赖域半径 Δ_k. 实际上，选取信赖域半径非常关键，它决定了算法的收敛性. 考虑到信赖域半径是 “对模型 $m_k(d)$ 相信的程度”，如果 $m_k(d)$ 对函数 $f(x)$ 近似较好，就应该扩大信赖域半径，在

更大的区域内使用这个近似，反之就应该减小信赖域半径重新计算. 我们引入如下定义来衡量 $m_k(d)$ 近似程度的好坏：

$$\rho_k = \frac{f(x^k) - f(x^k + d^k)}{m_k(0) - m_k(d^k)}, \tag{5.6.3}$$

其中 d^k 为求解子问题(5.6.2)得到的迭代方向. 根据 ρ_k 的定义我们知道，它是函数值实际下降量与预估下降量（即二阶近似模型下降量）的比值. 如果 ρ_k 接近于 1 ，说明用 $m_k(d)$ 来近似 $f(x)$ 是比较成功的，我们应该扩大 Δ_k；如果 ρ_k 非常小甚至为负，就说明我们过分地相信了二阶模型 $m_k(d)$，此时应该缩小 Δ_k. 使用这个机制可以动态调节 Δ_k，让二阶模型 $m_k(d)$ 的定义域处于一个合适的范围.

算法 5.7 信赖域算法

1. 给定最大半径 $\Delta_{\max}$，初始半径 Δ_0，初始点 x^0，$k \leftarrow 0$.
2. 给定参数 $0 \leqslant \eta < \bar{\rho}_1 < \bar{\rho}_2 < 1$，$\gamma_1 < 1 < \gamma_2$.
3. **while** 未达到收敛准则 **do**
4. 计算子问题(5.6.2)得到迭代方向 d^k.
5. 根据(5.6.3)式计算下降率 ρ_k.
6. 更新信赖域半径：

$$\Delta_{k+1} = \begin{cases} \gamma_1 \Delta_k, & \rho_k < \bar{\rho}_1, \\ \min\{\gamma_2 \Delta_k, \Delta_{\max}\}, & \rho_k > \bar{\rho}_2 \text{ 以及 } \|d^k\| = \Delta_k, \\ \Delta_k, & \text{其他}. \end{cases}$$

7. 更新自变量：

$$x^{k+1} = \begin{cases} x^k + d^k, & \rho_k > \eta, \\ x^k, & \text{其他}. \end{cases} \qquad /* \text{ 只有下降比例足够大才更新 } */$$

8. $k \leftarrow k + 1$.
9. **end while**

算法 5.7 给出完整的信赖域方法. 该算法虽然有一些参数，但是它对这些参数的取值并不敏感. 实际中可取 $\bar{\rho}_1 = 0.25, \bar{\rho}_2 = 0.75$ 以及 $\gamma_1 = 0.25, \gamma_2 = 2$. 注意，信赖域半径 Δ_k 不会无限增长，一是因为它有上界的控制，二是如果

信赖域约束不起作用（即二次模型最优值处于信赖域内），我们也无需增加信赖域半径. 只有当 $m_k(d)$ 近似足够好并且信赖域约束起作用时，才需要增加 Δ_k.

在算法 5.7 中只剩下一个关键问题没有说明：如何求解信赖域子问题(5.6.2)？下一个小节将给出两种方法.

5.6.2 信赖域子问题求解

在多数实际应用中，信赖域子问题(5.6.2)的解是无法显式写出的. 为了求出迭代方向 d^k，我们需要设计算法快速或近似求解子问题(5.6.2).

1. 迭代法

信赖域子问题是一个仅仅涉及二次函数的约束优化问题，那么能否用约束优化问题的最优性条件来求解子问题的解呢？下面的定理给出了子问题的最优解 p^* 需要满足的条件：

定理 5.10 *d^* 是信赖域子问题*

$$\min \quad m(d) = f + g^{\mathrm{T}}d + \frac{1}{2}d^{\mathrm{T}}Bd, \quad \text{s.t.} \quad \|d\| \leqslant \Delta \tag{5.6.4}$$

的全局极小解当且仅当 d^* 是可行的且存在 $\lambda \geqslant 0$ 使得

$$(B + \lambda I)d^* = -g, \tag{5.6.5a}$$

$$\lambda(\Delta - \|d^*\|) = 0, \tag{5.6.5b}$$

$$(B + \lambda I) \succeq 0. \tag{5.6.5c}$$

证明. 先证明必要性. 实际上，我们可以利用 KKT 条件来直接写出 d^* 所满足的关系. 问题(5.6.4)的拉格朗日函数为

$$L(d, \lambda) = f + g^{\mathrm{T}}d + \frac{1}{2}d^{\mathrm{T}}Bd - \frac{\lambda}{2}(\Delta^2 - \|d\|^2),$$

其中乘子 $\lambda \geqslant 0$. 根据 KKT 条件，d^* 是可行解，且

$$\nabla_d L(d^*, \lambda) = (B + \lambda I)d^* + g = 0.$$

此外还有互补条件

$$\frac{\lambda}{2}(\Delta^2 - \|d^*\|^2) = 0,$$

以上两式整理后就是(5.6.5a)式和(5.6.5b)式. 为了证明(5.6.5c)式，我们任取 d 满足 $\|d\| = \Delta$，根据最优性，有

$$m(d) \geqslant m(d^*) = m(d^*) + \frac{\lambda}{2}(\|d^*\|^2 - \|d\|^2).$$

利用(5.6.5a)式消去 g，代入上式整理可知

$$(d - d^*)^{\mathrm{T}}(B + \lambda I)(d - d^*) \geqslant 0,$$

由任意性可知 $B + \lambda I$ 半正定.

再证明充分性. 定义辅助函数

$$\hat{m}(d) = f + g^{\mathrm{T}}d + \frac{1}{2}d^{\mathrm{T}}(B + \lambda I)d = m(d) + \frac{\lambda}{2}d^{\mathrm{T}}d,$$

由条件 (5.6.5c) 可知 $\hat{m}(d)$ 关于 d 是凸函数. 根据条件 (5.6.5a)，d^* 满足凸函数一阶最优性条件，结合定理 4.4 可推出 d^* 是 $\hat{m}(d)$ 的全局极小值点，进而对任意可行解 d，我们有

$$m(d) \geqslant m(d^*) + \frac{\lambda}{2}(\|d^*\|^2 - \|d\|^2).$$

由互补条件 (5.6.5b) 可知 $\lambda(\Delta^2 - \|d^*\|^2) = 0$，代入上式消去 $\|d^*\|^2$ 得

$$m(d) \geqslant m(d^*) + \frac{\lambda}{2}(\Delta^2 - \|d\|^2) \geqslant m(d^*). \qquad \square$$

定理5.10提供了问题维数 n 较小时寻找 d^* 的一个方法. 根据 (5.6.5a) 式，最优解是以 λ 为参数的一族向量. 我们定义

$$d(\lambda) = -(B + \lambda I)^{-1}g, \tag{5.6.6}$$

则只需要寻找合适的 λ 使得(5.6.5b)式和(5.6.5c)式成立即可. 根据互补条件(5.6.5b)，当 $\lambda > 0$ 时必有 $\|d(\lambda)\| = \Delta$；根据半正定条件(5.6.5c)，λ 须大于等于 B 的最小特征值的相反数. 现在研究 $\|d(\lambda)\|$ 随 λ 变化的性质. 设 B 有特征值分解 $B = Q\Lambda Q^{\mathrm{T}}$，其中 $Q = [q_1, q_2, \cdots, q_n]$ 是正交矩阵，$\Lambda = \mathrm{Diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$ 是对角矩阵，$\lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n$ 是 B 的特征值. 为了方便，以下仅考虑 $\lambda_1 \leqslant 0$ 且 λ_1 是单特征根的情形. 其他情形可类似分析. 显然，$B + \lambda I$ 有特征值分解 $B + \lambda I = Q(\Lambda + \lambda I)Q^{\mathrm{T}}$. 对 $\lambda > -\lambda_1 \geqslant 0$，我们可直接写出 $d(\lambda)$ 的表达式:

$$d(\lambda) = -Q(\Lambda + \lambda I)^{-1}Q^{\mathrm{T}}g = -\sum_{i=1}^{n}\frac{q_i^{\mathrm{T}}g}{\lambda_i + \lambda}q_i. \tag{5.6.7}$$

这正是 $d(\lambda)$ 的正交分解，由正交性可容易求出

$$\|d(\lambda)\|^2 = \sum_{i=1}^{n} \frac{(q_i^{\mathrm{T}} g)^2}{(\lambda_i + \lambda)^2}. \tag{5.6.8}$$

根据(5.6.8)式可知当 $\lambda > -\lambda_1$ 且 $q_1^{\mathrm{T}} g \neq 0$ 时，$\|d(\lambda)\|^2$ 是关于 λ 的严格减函数，且有

$$\lim_{\lambda \to \infty} \|d(\lambda)\| = 0, \quad \lim_{\lambda \to -\lambda_1 +} \|d(\lambda)\| = +\infty.$$

根据连续函数介值定理，$\|d(\lambda)\| = \Delta$ 的解必存在且唯一. 一个典型的例子如图 5.13 所示.

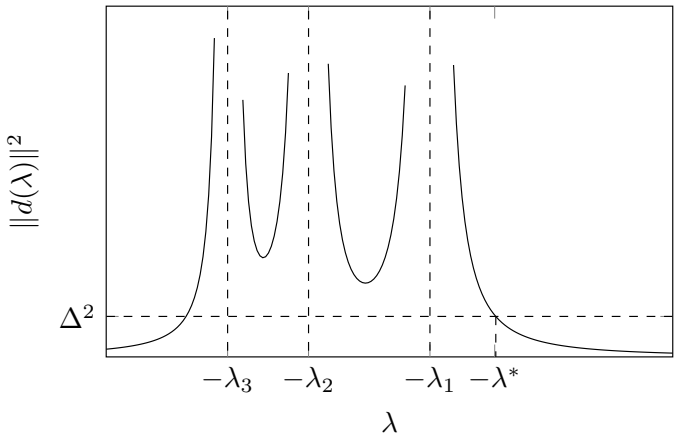


图 5.13 $\|d(\lambda)\|^2$ 随 λ 的变化关系

根据上面的分析，寻找 λ^* 已经转化为一个一元方程求根问题，我们可以使用牛顿法求解. 在得到最优解 λ^* 后，根据(5.6.5a)式即可求出迭代方向 d^*，这里略去细节，感兴趣的读者可参考 [98].

此外，在上面的分析中我们假定了 $q_1^{\mathrm{T}} g \neq 0$，在实际中这个条件未必满足. 当 $q_1^{\mathrm{T}} g = 0$ 时，(5.6.8)式将没有和 λ_1 相关的项. 此时未必存在 $\lambda^* > -\lambda_1$ 使得 $\|d(\lambda^*)\| = \Delta$ 成立. 记 $M = \lim\limits_{\lambda \to -\lambda_1 +} \|d(\lambda)\|$，当 $M \geqslant \Delta$ 时，仍然可以根据介值定理得出 $\lambda^* (> -\lambda_1)$ 的存在性；而当 $M < \Delta$ 时，无法利用前面的分析求出 λ^* 和 d^*，此时信赖域子问题变得比较复杂. 实际上，$q_1^{\mathrm{T}} g = 0$ 且 $M < \Delta$ 的情形被人们称为 “困难情形 (hard case)”. 此情形发生时，区间 $(-\lambda_1, +\infty)$ 中的点无法使得 (5.6.5b) 成立，而定理 5.10 的结果说明 $\lambda^* \in [-\lambda_1, +\infty)$，因此必有 $\lambda^* = -\lambda_1$.

为了求出 d^*，可以利用（奇异）线性方程组 (5.6.5a) 解的结构，其通解可以写为

$$d(\alpha) = -\sum_{i=2}^{n} \frac{q_i^{\mathrm{T}} g}{\lambda_i - \lambda_1} q_i + \alpha q_1, \quad \alpha \in \mathbb{R}.$$

由正交性，

$$\|d(\alpha)\|^2 = \alpha^2 + \sum_{i=2}^{n} \frac{(q_i^{\mathrm{T}} g)^2}{(\lambda_i - \lambda_1)^2}.$$

注意在困难情形中有 $M = \sqrt{\sum_{i=2}^{n} \frac{(q_i^{\mathrm{T}} g)^2}{(\lambda_i - \lambda_1)^2}} < \Delta$，因此必存在 α^* 使得 $\|d(\alpha^*)\| = \Delta$. 这就求出了 d^* 的表达式.

2. 截断共轭梯度法

我们再介绍一种信赖域子问题的求解方法. 既然问题(5.6.2)的解不易求出，能否写出它的一个近似解呢？Steihaug [117] 在 1983 年对共轭梯度法进行了改造，使其成为能求解问题(5.6.2)的算法. 此算法能够应用在大规模问题中，是一种非常有效的信赖域子问题的求解方法. 我们知道，子问题(5.6.2)和一般的二次极小化问题相差一个约束，如果先不考虑其中的约束 $\|d\| \leqslant \Delta$ 而直接使用共轭梯度法求解，在迭代过程中应该能找到合适的迭代点作为信赖域子问题的近似解. 这就是截断共轭梯度法的基本思想.

为了介绍截断共轭梯度法，我们简要回顾一下标准共轭梯度法的迭代过程. 对于二次极小化问题

$$\min_{s} \quad q(s) \xlongequal{\text{def}} g^{\mathrm{T}} s + \frac{1}{2} s^{\mathrm{T}} B s,$$

给定初值 $s^0 = 0, r^0 = g, p^0 = -g$，共轭梯度法的迭代过程为

$$\begin{aligned}
\alpha_{k+1} &= \frac{\|r^k\|^2}{(p^k)^{\mathrm{T}} B p^k}, \\
s^{k+1} &= s^k + \alpha_k p^k, \\
r^{k+1} &= r^k + \alpha_k B p^k, \\
\beta_k &= \frac{\|r^{k+1}\|^2}{\|r^k\|^2}, \\
p^{k+1} &= -r^{k+1} + \beta_k p^k,
\end{aligned}$$

其中迭代序列 $\{s^k\}$ 最终的输出即为二次极小化问题的解，算法的终止准则是判断 $\|r^k\|$ 是否足够小. 截断共轭梯度法则是给标准的共轭梯度法增加了

两条终止准则，并对最后一步的迭代点 s^k 进行修正来得到信赖域子问题的解. 考虑到矩阵 B 不一定是正定矩阵，在迭代过程中可能会产生如下三种情况：

(1) $(p^k)^{\mathrm{T}}Bp^k \leqslant 0$，即 B 不是正定矩阵. 我们知道共轭梯度法不能处理非正定的线性方程组，遇到这种情况应该立即终止算法. 但根据这个条件也找到了一个负曲率方向，此时只需要沿着这个方向走到信赖域边界即可.

(2) $(p^k)^{\mathrm{T}}Bp^k > 0$ 但 $\|s^{k+1}\| \geqslant \Delta$，这表示若继续进行共轭梯度法迭代，则点 s^{k+1} 将处于信赖域之外或边界上，此时必须马上停止迭代，并在 s^k 和 s^{k+1} 之间找一个近似解.

(3) $(p^k)^{\mathrm{T}}Bp^k > 0$ 且 $\|r^{k+1}\|$ 充分小，这表示若共轭梯度法成功收敛到信赖域内. 子问题(5.6.2)和不带约束的二次极小化问题是等价的.

从上述终止条件来看截断共轭梯度法仅仅产生了共轭梯度法的部分迭代点，这也是该方法名字的由来.

算法5.8给出截断共轭梯度法的迭代过程，其中的三个判断分别对应了之前叙述的三种情况. 注意，在不加限制时，下一步迭代点总是为 $s^{k+1} = s^k + \alpha_k p^k$. 当情况 (1) 发生时，只需要沿着 p^k 走到信赖域边界；当情况 (2) 发生时，由于 $\|s^k\| < \Delta, \|s^k + \alpha_k p^k\| \geqslant \Delta$，由连续函数介值定理可得 τ 必定存在且处于区间 $(0, \alpha_k]$ 内.

5.6.3 收敛性分析

本小节简要介绍信赖域算法的收敛性结果. 我们将着重于介绍定理的条件和最终结论，而略去较为烦琐的证明部分.

1. 柯西点

为了估计求解每个信赖域子问题得到的函数值改善情况，我们引入柯西点的定义.

定义 5.6 (柯西点) 设 $m_k(d)$ 是 $f(x)$ 在点 $x = x^k$ 处的二阶近似，常数 τ_k 为如下优化问题的解：

$$\begin{aligned} \min \quad & m_k(-\tau \nabla f(x^k)), \\ \text{s.t.} \quad & \|\tau \nabla f(x^k)\| \leqslant \Delta_k, \tau \geqslant 0. \end{aligned}$$

算法 5.8 截断共轭梯度法 (Steihaug-CG)

1. 给定精度 $\varepsilon > 0$，初始化 $s^0 = 0, r^0 = g, p^0 = -g$，$k \leftarrow 0$.
2. **if** $\|p^0\| \leqslant \varepsilon$ **then**
3. 算法停止，输出 $s = 0$.
4. **end if**
5. **loop**
6. **if** $(p^k)^{\mathrm{T}} B p^k \leqslant 0$ **then**
7. 计算 $\tau > 0$ 使得 $\|s^k + \tau p^k\| = \Delta$.
8. 算法停止，输出 $s = s^k + \tau p^k$.
9. **end if**
10. 计算 $\alpha_k = \dfrac{\|r^k\|^2}{(p^k)^{\mathrm{T}} B p^k}$，更新 $s^{k+1} = s^k + \alpha_k p^k$.
11. **if** $\|s^{k+1}\| \geqslant \Delta$ **then**
12. 计算 $\tau > 0$ 使得 $\|s^k + \tau p^k\| = \Delta$.
13. 算法停止，输出 $s = s^k + \tau p^k$.
14. **end if**
15. 计算 $r^{k+1} = r^k + \alpha_k B p^k$.
16. **if** $\|r^{k+1}\| < \varepsilon \|r^0\|$ **then**
17. 算法停止，输出 $s = s^{k+1}$.
18. **end if**
19. 计算 $\beta_k = \dfrac{\|r^{k+1}\|^2}{\|r^k\|^2}$，更新 $p^{k+1} = -r^{k+1} + \beta_k p^k$.
20. $k \leftarrow k + 1$.
21. **end loop**

则称 $x_{\mathrm{C}}^k \overset{\mathrm{def}}{=\!=} x^k + d_{\mathrm{C}}^k$ 为**柯西点**，其中 $d_{\mathrm{C}}^k = -\tau_k \nabla f(x^k)$.

根据柯西点的定义，它实际上是对 $m_k(d)$ 进行了一次精确线搜索的梯度法，不过这个线搜索是考虑了信赖域约束的. 图5.14直观地解释了柯西点的含义.

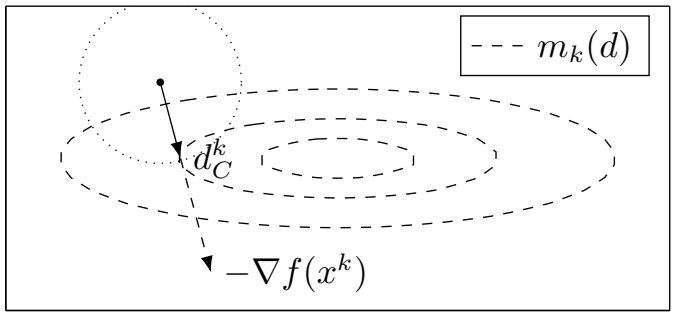


图 5.14 柯西点

实际上，给定 $m_k(d)$，柯西点可以显式计算出来. 为了方便我们用 g^k 表示 $\nabla f(x^k)$，根据 τ_k 的定义，容易计算出其表达式为

$$\tau_k = \begin{cases} \dfrac{\Delta_k}{\|g^k\|}, & (g^k)^{\mathrm{T}} B^k g^k \leqslant 0, \\ \min\left\{ \dfrac{\|g^k\|^2}{(g^k)^{\mathrm{T}} B^k g^k}, \dfrac{\Delta_k}{\|g^k\|} \right\}, & \text{其他.} \end{cases}$$

以上的分析表明，柯西点是信赖域子问题(5.6.2)的一个可行点，但在实际计算中人们一般不会将柯西点作为下一步迭代点的近似. 这是由于求柯西点本质上为一个带截断步长的最速下降法，它并没有充分利用海瑟矩阵 B^k 的信息. 即便如此，人们还是可以将柯西点作为信赖域子问题算法的一个评判标准，即要求子问题算法产生的迭代点至少比柯西点要好. 而容易看出，若迭代点取为柯西点，二次模型的目标函数值依然是下降的. 实际上，我们有如下引理（证明见 [98]引理 4.3）：

引理 5.2 (柯西点的下降量) 设 d_C^k 为求解柯西点产生的下降方向，则

$$m_k(0) - m_k(d_C^k) \geqslant \frac{1}{2}\|g^k\| \min\left\{ \Delta_k, \frac{\|g^k\|}{\|B^k\|_2} \right\}. \tag{5.6.9}$$

而前一小节介绍的子问题求解方法（如迭代法，截断共轭梯度法，Dogleg 方法）得到的迭代方向 d^k 均满足

$$m_k(0) - m_k(d^k) \geqslant c_2(m_k(0) - m_k(d_C^k)).$$

这也就意味着估计式

$$m_k(0) - m_k(d^k) \geqslant \frac{1}{2}c_2\|g^k\| \min\left\{ \Delta_k, \frac{\|g^k\|}{\|B^k\|_2} \right\} \tag{5.6.10}$$

在很多情况下都会成立. 这为我们证明信赖域算法的收敛性提供了基础.

2. 全局收敛性

现在介绍信赖域算法的全局收敛性. 回顾信赖域算法5.7, 我们引入了一个参数 η 来确定是否应该更新迭代点. 这分为两种情况: 当 $\eta = 0$ 时, 只要原目标函数有下降量就接受信赖域迭代步的更新; 当 $\eta > 0$ 时, 只有当改善量 ρ_k 达到一定程度时再进行更新. 在这两种情况下得到的收敛性结果是不同的, 我们分别介绍这两种结果.

根据 [98]定理 4.5, 在 $\eta = 0$ 的条件下有如下收敛性定理:

定理 5.11 (全局收敛性 1) *设近似海瑟矩阵 B^k 有界, 即 $\|B^k\|_2 \leqslant M, \forall\, k$, $f(x)$ 在下水平集 $\mathcal{L} = \{x \mid f(x) \leqslant f(x^0)\}$ 上有下界, 且 $\nabla f(x)$ 在 $\mathcal{L}$ 的一个开邻域内利普希茨连续. 若 d^k 为信赖域子问题的近似解且满足(5.6.10)式, 算法5.7选取参数 $\eta = 0$, 则*

$$\liminf_{k\to\infty} \|\nabla f(x^k)\| = 0,$$

即 x^k 的聚点中包含稳定点.

定理 5.11 表明若无条件接受信赖域子问题的更新, 则算法 5.7 仅仅有子序列的收敛性, 迭代点序列本身不一定收敛. 根据 [98]定理 4.6, 下面的定理则说明选取 $\eta > 0$ 可以改善收敛性结果.

定理 5.12 (全局收敛性 2) *在定理 5.11的条件下, 若算法5.7选取参数 $\eta > 0$, 且信赖域子问题近似解 d^k 满足(5.6.10)式, 则*

$$\lim_{k\to\infty} \|\nabla f(x^k)\| = 0.$$

和牛顿类算法不同, 信赖域算法具有全局收敛性, 因此它对迭代初值选取的要求比较弱. 而牛顿法的收敛性极大地依赖初值的选取.

3. 局部收敛性

再来介绍信赖域算法的局部收敛性. 在构造信赖域子问题时利用了 $f(x)$ 的二阶信息, 它在最优点附近应该具有牛顿法的性质. 特别地, 当近似矩阵 B^k 取为海瑟矩阵 $\nabla^2 f(x^k)$ 时, 根据信赖域子问题的更新方式, 二次模型 $m_k(d)$ 将会越来越逼近原函数 $f(x)$, 最终信赖域约束 $\|d\| \leqslant \Delta_k$ 将会失效. 此

时信赖域方法将会和牛顿法十分接近，而根据定理5.6，牛顿法有 Q-二次收敛的性质，这个性质很自然地会继承到信赖域算法上.

和牛顿法不同的是，在信赖域迭代算法中，我们并不知道迭代点列是否已经接近最优点. 即使信赖域半径约束已经不起作用，如果 B^k 没有取为海瑟矩阵 $\nabla^2 f(x^k)$ 或者信赖域子问题没有精确求解，d^k 一般也不会等于牛顿方向 d_N^k，但是它们的误差往往是越来越小的. 根据 [98]定理 4.9，我们有如下定理：

定理 5.13 设 $f(x)$ 在最优点 $x = x^*$ 的一个邻域内二阶连续可微，且 $\nabla f(x)$ 利普希茨连续，在最优点 x^* 处二阶充分条件成立，即 $\nabla^2 f(x^*) \succ 0$. 若迭代点列 $\{x^k\}$ 收敛到 x^*，且在迭代中选取 B^k 为海瑟矩阵 $\nabla^2 f(x^k)$，则对充分大的 k，任意满足(5.6.10)式的信赖域子问题算法产生的迭代方向 d^k 均满足

$$\|d^k - d_N^k\| = o(\|d_N^k\|), \tag{5.6.11}$$

其中 d_N^k 为第 k 步迭代的牛顿方向且满足假设 $\|d_N^k\| \leqslant \dfrac{\Delta_k}{2}$.

定理 5.13 说明若信赖域算法收敛，则当 k 充分大时，信赖域半径的约束终将失效，且算法产生的迭代方向将会越来越接近牛顿方向.

根据定理 5.13 很容易得到收敛速度的估计.

推论 5.3 (信赖域算法的局部收敛速度) 在定理5.13的条件下，信赖域算法产生的迭代序列 $\{x^k\}$ 具有 Q-超线性收敛速度.

证明. 根据定理5.6，对牛顿方向，

$$\|x^k + d_N^k - x^*\| = \mathcal{O}(\|x^k - x^*\|^2),$$

因此得到估计 $\|d_N^k\| = \mathcal{O}(\|x^k - x^*\|)$. 又根据定理5.13，

$$\|x^k + d^k - x^*\| \leqslant \|x^k + d_N^k - x^*\| + \|d^k - d_N^k\| = o(\|x^k - x^*\|).$$

这说明信赖域算法是 Q-超线性收敛的. □

容易看出，若在迭代后期 $d^k = d_N^k$ 能得到满足，则信赖域算法是 Q-二次收敛的. 很多算法都会有这样的性质，例如前面提到的截断共轭梯度法和 Dogleg 方法. 因此在实际应用中，截断共轭梯度法是最常用的信赖域子问题的求解方法，使用此方法能够同时兼顾全局收敛性和局部 Q-二次收敛性.

5.6.4 应用举例

考虑逻辑回归问题

$$\min_{x} \quad \frac{1}{m}\sum_{i=1}^{m}\ln(1+\exp(-b_i a_i^{\mathrm{T}} x)) + \lambda\|x\|_2^2, \tag{5.6.12}$$

这里选取 $\lambda = \dfrac{1}{100m}$. 其导数和海瑟矩阵计算参见第 5.4 节. 同样地，我们选取 LIBSVM 上的数据集，调用信赖域算法求解代入数据集后的问题(5.6.12)，其迭代收敛过程见图5.15. 其中使用截断共轭梯度法来求解信赖域子问题，精度设置同第 5.4 节的牛顿法一致. 从图中可以看到，在精确解附近梯度范数具有 Q-超线性收敛性质. 由于这个问题是强凸的，所以选取一个较大的初始信赖域半径 ($\sqrt{n}$). 在数据集 a9a 和 ijcnn1 的求解中，信赖域子问题的求解没有因为超出信赖域边界而停机，因此和第 5.4 节中牛顿法的数值表现一致.

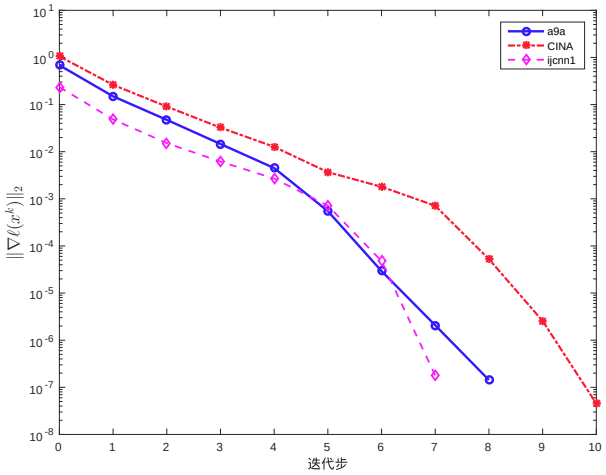


图 5.15 逻辑回归问题

5.7 非线性最小二乘问题算法

本节研究非线性最小二乘问题的算法. 非线性最小二乘问题是一类特殊的无约束优化问题，它有非常广泛的实际应用背景. 例如在统计中，我们经常建立如下带参数的模型：

$$b = \phi(a;x) + \varepsilon,$$

其中 a 为自变量，b 为响应变量，它们之间的关系由函数 $\phi(\cdot;x)$ 决定且 x 是参数，ε 是噪声项，即观测都是有误差的. 我们的目的是要根据观测 (a_i,b_i)，估计未知参数 x 的值. 若 ε 服从高斯分布，则使用 ℓ_2 范数平方是处理高斯噪声最好的方式：对 ℓ_2 范数平方损失函数

$$f(x)=\frac{1}{m}\sum_{i=1}^{m}\|b_i-\phi(a_i;x)\|^2$$

进行极小化即可求出未知参数 x 的估计. 而对 ℓ_2 范数平方损失函数求解极小值就是一个最小二乘问题.

最小二乘问题一般属于无约束优化问题，但由于问题特殊性，人们针对其结构设计了许多算法快速求解. 一般地，设 x^* 为最小二乘问题的解，根据最优解处残量 $\sum_{i=1}^{m}\|b_i-\phi(a_i;x)\|^2$ 的大小，可以将最小二乘问题分为**小残量问题**和**大残量问题**. 本节针对小残量问题介绍两种方法：高斯 – 牛顿算法和 LM 方法.

5.7.1 非线性最小二乘问题

考虑非线性最小二乘问题的一般形式

$$f(x)=\frac{1}{2}\sum_{j=1}^{m}r_j^2(x), \tag{5.7.1}$$

其中 $r_j:\mathbb{R}^n\to\mathbb{R}$ 是光滑函数，并且假设 $m\geqslant n$. 我们称 r_j 为残差. 为了表述问题的方便，定义残差向量 $r:\mathbb{R}^n\to\mathbb{R}^m$ 为

$$r(x)=(r_1(x),r_2(x),\cdots,r_m(x))^{\mathrm{T}}.$$

使用这一定义，函数 $f(x)$ 可以写为 $f(x)=\frac{1}{2}\|r(x)\|_2^2$.

问题(5.7.1)是一个无约束优化问题，可以使用前面讲过的任何一种算法求解. 为此我们直接给出 $f(x)$ 的梯度和海瑟矩阵：

$$\nabla f(x)=J(x)^{\mathrm{T}}r(x), \tag{5.7.2a}$$

$$\nabla^2 f(x)=J(x)^{\mathrm{T}}J(x)+\sum_{i=1}^{m}r_i(x)\nabla^2 r_i(x), \tag{5.7.2b}$$

其中 $J(x)\in\mathbb{R}^{m\times n}$ 是向量值函数 $r(x)$ 在点 x 处的雅可比矩阵. 这里指出，$\nabla^2 f(x)$ 在形式上分为两部分，分别为 $J(x)^{\mathrm{T}}J(x)$ 和 $\sum_{i=1}^{m}r_i(x)\nabla^2 r_i(x)$. 处理这

两部分的难度是截然不同的：注意到计算 $\nabla f(x)$ 时需要 $r(x)$ 的雅可比矩阵，因此海瑟矩阵的前一项是自然得到的，不需要进行额外计算；而海瑟矩阵的第二项则需要计算每个 $\nabla^2 r_i(x)$，这会导致额外计算量，因此很多最小二乘算法就是根据这个性质来设计的.

5.7.2 高斯 – 牛顿算法

高斯 – 牛顿法是求解非线性最小二乘问题的经典方法，它可以看成是结合了线搜索的牛顿法的变形. 既然海瑟矩阵中有关 $r_i(x)$ 的二阶导数项不易求出，高斯 – 牛顿法不去计算这一部分，直接使用 $J(x)^{\mathrm{T}}J(x)$ 作为海瑟矩阵的近似矩阵来求解牛顿方程. 我们用 J^k 简记 $J(x^k)$. 高斯 – 牛顿法产生的下降方向 d^k 满足

$$(J^k)^{\mathrm{T}}J^k d^k = -(J^k)^{\mathrm{T}}r^k. \tag{5.7.3}$$

方程(5.7.3)正是法方程的形式，而由线性代数的知识可知，不管 J^k 是否是满秩矩阵，方程 (5.7.3)一定存在解. 实际上，该方程是如下线性最小二乘问题的最优性条件：

$$\min_d \quad \frac{1}{2}\|J^k d + r^k\|^2.$$

在求解线性最小二乘问题时，我们只需要对 J^k 做 QR 分解，因此矩阵 $(J^k)^{\mathrm{T}}J^k$ 无需计算出来.

高斯 – 牛顿法的框架如算法5.9. 为了方便理解，我们将求解线性最小二乘问题的方法进行了展开. 高斯 – 牛顿法每一步的运算量是来自残差向量

算法 5.9 高斯 – 牛顿法

1. 给定初始值 x^0，$k \leftarrow 0$.
2. **while** 未达到收敛准则 **do**
3. 计算残差向量 r^k，雅可比矩阵 J^k.
4. 计算 J^k 的 QR 分解：$J^k = Q^k R^k$，其中 $Q^k \in \mathbb{R}^{m\times n}, R^k \in \mathbb{R}^{n\times n}$.
5. 求解方程 $R^k d^k = -(Q^k)^{\mathrm{T}} r^k$ 得下降方向 d^k.
6. 使用线搜索准则计算步长 α_k.
7. 更新：$x^{k+1} = x^k + \alpha_k d^k$.
8. $k \leftarrow k+1$.
9. **end while**

r^k 和雅可比矩阵 J^k，和其他算法相比，它的计算量较小. 我们还注意到，若

J^k 是满秩矩阵，则高斯 – 牛顿法得到的方向 d^k 总是一个下降方向，这是因为

$$(d^k)^{\mathrm{T}}\nabla f(x^k) = (d^k)^{\mathrm{T}}(J^k)^{\mathrm{T}}r^k = -\|J^k d^k\|^2 < 0.$$

这也是高斯 – 牛顿法的优点．在此之前我们介绍了牛顿法，但它并不总是保证 d^k 是下降方向．而高斯 – 牛顿法使用一个半正定矩阵来近似牛顿矩阵，可以获得较好的下降方向．

一个很自然的问题是：高斯 – 牛顿法使用了近似矩阵来求解牛顿方程，那么在什么情况下这个近似是合理的？直观上看，根据海瑟矩阵(5.7.2b)的表达式，当 $(J^k)^{\mathrm{T}}J^k$ 这一部分起主导时，所使用的近似是有意义的．一个充分条件就是在最优点 x^* 处 $r_i(x^*)$ 的值都很小．此时高斯 – 牛顿法和牛顿法相近，它们也有很多相似的性质．如果残差向量 $r(x^*)$ 模长较大，则仅仅使用 $(J^k)^{\mathrm{T}}J^k$ 并不能很好地近似 $\nabla^2 f(x^k)$，此时高斯 – 牛顿法可能收敛很慢甚至发散．

接下来给出高斯 – 牛顿法的收敛性质．通过上面的描述可以注意到，雅可比矩阵 J^k 的非奇异性是一个很关键的因素，因此我们在这个条件下建立收敛性．假设雅可比矩阵 $J(x)$ 的奇异值一致地大于 0，即存在 $\gamma > 0$ 使得

$$\|J(x)z\| \geqslant \gamma\|z\|, \quad \forall\, x \in \mathcal{N}, \quad \forall\, z \in \mathbb{R}^n, \tag{5.7.4}$$

其中 $\mathcal{N}$ 是下水平集

$$\mathcal{L} = \{x \mid f(x) \leqslant f(x^0)\} \tag{5.7.5}$$

的一个邻域，x^0 是算法的初始点，且假设 $\mathcal{L}$ 是有界的．

在前面的假设下，有如下收敛性定理：

定理 5.14 (全局收敛性) *如果每个残差函数 r_j 在有界下水平集(5.7.5)的一个邻域 $\mathcal{N}$ 内是利普希茨连续可微的，并且雅可比矩阵 $J(x)$ 在 $\mathcal{N}$ 内满足一致满秩条件(5.7.4)，而步长满足 Wolfe 准则(5.1.4)，则对高斯 – 牛顿法得到的序列 $\{x^k\}$ 有*

$$\lim_{k\to\infty}(J^k)^{\mathrm{T}}r^k = 0.$$

证明. 这里直接验证 Zoutendijk 条件 (5.1.7) 成立即可．首先，选择有界下水平集 $\mathcal{L}$ 的邻域 $\mathcal{N}$ 足够小，从而使得存在 $L > 0, \beta > 0$，对于任何 $x, \tilde{x} \in \mathcal{N}$ 以及任意的 $j = 1, 2, \cdots, m$，以下条件被满足：

$$\|r_j(x)\| \leqslant \beta, \qquad \|\nabla r_j(x)\| \leqslant \beta, \tag{5.7.6}$$

$$|r_j(x) - r_j(\tilde{x})| \leqslant L\|x - \tilde{x}\|, \qquad \|\nabla r_j(x) - \nabla r_j(\tilde{x})\| \leqslant L\|x - \tilde{x}\|. \tag{5.7.7}$$

易得对任意的 $x \in \mathcal{L}$ 存在 $\tilde{\beta}$ 使得 $\|J(x)\| = \|J(x)^{\mathrm{T}}\| \leqslant \tilde{\beta}$，及 $\nabla f(x) = J(x)^{\mathrm{T}} r(x)$ 是利普希茨连续函数. 记 θ_k 是高斯 – 牛顿方向 d^k 与负梯度方向的夹角，则

$$\cos\theta_k = -\frac{\nabla f(x^k)^{\mathrm{T}} d^k}{\|d^k\|\|\nabla f(x^k)\|} = \frac{\|J^k d^k\|^2}{\|d^k\|\|(J^k)^{\mathrm{T}} J^k d^k\|} \geqslant \frac{\gamma^2\|d^k\|^2}{\tilde{\beta}^2\|d^k\|^2} = \frac{\gamma^2}{\tilde{\beta}^2} > 0.$$

根据推论 5.1 即可得 $\nabla f(x^k) \to 0$. □

定理5.14的关键假设是一致满秩条件 (5.7.4). 实际上，若 J^k 不满秩，则线性方程组 (5.7.3)有无穷多个解. 如果对解的性质不提额外要求，则无法推出 $\cos\theta_k$ 一致地大于零. 此时收敛性可能不成立.

当 $(J^k)^{\mathrm{T}} J^k$ 在海瑟矩阵 (5.7.2b) 中占据主导部分时，高斯 – 牛顿算法可能会有更快的收敛速度. 类似于牛顿法，我们给出高斯 – 牛顿法的局部收敛性.

定理 5.15 (局部收敛性) 设 $r_i(x)$ 二阶连续可微，x^* 是最小二乘问题 (5.7.1) 的最优解，海瑟矩阵 $\nabla^2 f(x)$ 和其近似矩阵 $J(x)^{\mathrm{T}} J(x)$ 均在点 x^* 的一个邻域内利普希茨连续，则当高斯 – 牛顿算法步长 α_k 恒为 1 时，

$$\|x^{k+1} - x^*\| \leqslant C\|((J^*)^{\mathrm{T}} J^*)^{-1} H^*\|\|x^k - x^*\| + \mathcal{O}(\|x^k - x^*\|^2), \tag{5.7.8}$$

其中 $H^* = \sum_{i=1}^{m} r_i(x^*)\nabla^2 r_i(x^*)$ 为海瑟矩阵 $\nabla^2 f(x^*)$ 去掉 $J(x^*)^{\mathrm{T}} J(x^*)$ 的部分，$C > 0$ 为常数.

证明. 根据迭代公式，

$$\begin{aligned} x^{k+1} - x^* &= x^k + d^k - x^* \\ &= ((J^k)^{\mathrm{T}} J^k)^{-1}((J^k)^{\mathrm{T}} J^k(x^k - x^*) + \nabla f(x^*) - \nabla f(x^k)). \end{aligned} \tag{5.7.9}$$

由泰勒展开，

$$\begin{aligned} \nabla f(x^k) - \nabla f(x^*) = &\int_0^1 J^{\mathrm{T}} J(x^* + t(x^k - x^*))(x^k - x^*)\mathrm{d}t + \\ &\int_0^1 H(x^* + t(x^k - x^*))(x^k - x^*)\mathrm{d}t, \end{aligned}$$

其中 $J^{\mathrm{T}} J(x)$ 是 $J^{\mathrm{T}}(x)J(x)$ 的简写，$H(x) = \nabla^2 f(x) - J^{\mathrm{T}} J(x)$ 为海瑟矩阵剩余

部分. 将泰勒展开式代入(5.7.9)式右边，取范数进行估计，有

$$\begin{aligned}
&\|(J^k)^{\mathrm{T}}J^k(x^k-x^*)+\nabla f(x^*)-\nabla f(x^k)\| \\
\leqslant&\int_0^1\|(J^{\mathrm{T}}J(x^k)-J^{\mathrm{T}}J(x^*+t(x^k-x^*)))(x^k-x^*)\|\mathrm{d}t+ \\
&\int_0^1\|H(x^*+t(x^k-x^*))(x^k-x^*)\|\mathrm{d}t \\
\leqslant&\frac{L}{2}\|x^k-x^*\|^2+C\|H^*\|\|x^k-x^*\|,
\end{aligned}$$

其中 L 是 $J^{\mathrm{T}}J(x)$ 的利普希茨常数. 最后一个不等式是因为我们使用 H^* 来近似 $H(x^*+t(x^k-x^*))$，由连续性，存在 $C>0$ 以及点 x^* 的一个邻域 $\mathcal{N}$，对任意的 $x\in\mathcal{N}$ 有 $\|H(x)\|\leqslant C\|H(x^*)\|$. 将上述估计代入 (5.7.9)式即可得到(5.7.8)式. □

定理5.15指出，如果 $\|H(x^*)\|$ 充分小，则高斯 – 牛顿法可以达到 Q-线性收敛速度；而当 $\|H(x^*)\|=0$ 时，收敛速度是 Q-二次的. 如果 $\|H(x^*)\|$ 较大，则高斯 – 牛顿法很可能会失效.

5.7.3 Levenberg-Marquardt 方法

1. 信赖域型 LM 方法

Levenberg-Marquardt (LM) 方法是由 Levenberg 在 1944 年提出的求解非线性最小二乘问题的方法 [77]. 它本质上是一种信赖域型方法，主要应用场合是当矩阵 $(J^k)^{\mathrm{T}}J^k$ 奇异时，它仍然能给出一个下降方向. LM 方法每一步求解如下子问题：

$$\min_d\quad\frac{1}{2}\|J^kd+r^k\|^2,\quad\text{s.t.}\quad\|d\|\leqslant\Delta_k. \tag{5.7.10}$$

事实上，LM 方法将如下近似当作信赖域方法中的 m_k:

$$m_k(d)=\frac{1}{2}\|r^k\|^2+d^{\mathrm{T}}(J^k)^{\mathrm{T}}r^k+\frac{1}{2}d^{\mathrm{T}}(J^k)^{\mathrm{T}}J^kd. \tag{5.7.11}$$

该方法使用 $B^k=(J^k)^{\mathrm{T}}J^k$ 来近似海瑟矩阵，这个取法是从高斯 – 牛顿法推广而来的. LM 方法并不直接使用海瑟矩阵来求解. 以下为了方便，省去迭代指标 k. 子问题(5.7.10)是信赖域子问题，上一节讨论过这个子问题的一些好的性质. 根据定理5.10 ，可直接得到如下推论：

推论 5.4 向量 d^* 是信赖域子问题

$$\min_d \quad \frac{1}{2}\|Jd+r\|^2, \quad \text{s.t.} \quad \|d\| \leqslant \Delta$$

的解当且仅当 d^* 是可行解并且存在数 $\lambda \geqslant 0$ 使得

$$(J^{\mathrm{T}}J+\lambda I)d^* = -J^{\mathrm{T}}r, \tag{5.7.12}$$

$$\lambda(\Delta - \|d^*\|) = 0. \tag{5.7.13}$$

注意到 $J^{\mathrm{T}}J$ 是半正定矩阵，因此条件(5.6.5c)是自然成立的.

下面简要说明如何求解 LM 子问题(5.7.10). 实际上，和信赖域子问题中的迭代法相同，我们先通过求根的方式来确定 λ 的选取，然后直接求得 LM 方程的迭代方向. 由于 LM 子问题的特殊性，可以不显式求出矩阵 $J^{\mathrm{T}}J+\lambda I$ 的 Cholesky 分解，而仍然是借助 QR 分解，进而无需算出 $J^{\mathrm{T}}J+\lambda I$. 注意，$(J^{\mathrm{T}}J+\lambda I)d = -J^{\mathrm{T}}r$ 实际上是最小二乘问题

$$\min_d \quad \left\| \begin{bmatrix} J \\ \sqrt{\lambda} I \end{bmatrix} d + \begin{bmatrix} r \end{bmatrix} \right\| \tag{5.7.14}$$

的最优性条件. 此问题的系数矩阵带有一定结构，每次改变 λ 进行试探时，有关 J 的块是不变的，因此无需重复计算 J 的 QR 分解. 具体来说，设 $J=QR$ 为 J 的 QR 分解，其中 $Q \in \mathbb{R}^{m\times n}, R \in \mathbb{R}^{n\times n}$. 我们有

$$\begin{bmatrix} J \\ \sqrt{\lambda} I \end{bmatrix} = \begin{bmatrix} QR \\ \sqrt{\lambda} I \end{bmatrix} = \begin{bmatrix} Q & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} R \\ \sqrt{\lambda} I \end{bmatrix}.$$

矩阵 $\begin{bmatrix} R \\ \sqrt{\lambda} I \end{bmatrix}$ 含有较多零元素，利用这个特点我们可以使用 Householder 变换或 Givens 变换来完成此矩阵的 QR 分解. 如果矩阵 J 没有显式形式，只能提供矩阵乘法，则仍然可以用截断共轭梯度法，即算法5.8来求解子问题(5.7.10).

LM 方法的收敛性也可以直接从信赖域方法的收敛性得出，我们直接给出收敛性定理.

定理 5.16 假设常数 $\eta \in \left(0, \frac{1}{4}\right)$，下水平集 $\mathcal{L}$ 是有界的且每个 $r_i(x)$ 在下水平集 $\mathcal{L}$ 的一个邻域 $\mathcal{N}$ 内是利普希茨连续可微的. 假设对于任意的 k，子问题(5.7.10)的近似解 d_k 满足

$$m_k(0) - m_k(d^k) \geqslant c_1 \|(J^k)^{\mathrm{T}} r^k\| \min\left\{\Delta_k, \frac{\|(J^k)^{\mathrm{T}} r^k\|}{\|(J^k)^{\mathrm{T}} J^k\|}\right\},$$

其中 $c_1 > 0$ 且 $\|d^k\| \leqslant \gamma\Delta_k, \gamma \geqslant 1$，则

$$\lim_{k\to\infty} \nabla f(x^k) = \lim_{k\to\infty} (J^k)^{\mathrm{T}} r^k = 0.$$

2. LMF 方法

信赖域型 LM 方法本质上是固定信赖域半径 Δ，通过迭代寻找满足条件(5.7.12)的乘子 λ，每一步迭代需要求解线性方程组

$$(J^{\mathrm{T}}J + \lambda I)d = -J^{\mathrm{T}}r.$$

这个求解过程在 LM 方法中会占据相当大的计算量，能否简化这个计算呢? 在学习信赖域子问题求解方法时，我们仔细讨论了迭代法，图5.13表明，当 $\lambda > -\lambda_1$ 时，下降方向 d 的模长随着 λ 的增加而减小. 在 LM 方法中，一个显然的结论就是 $-\lambda_1 < 0$. 这就意味着 Δ 的大小被 λ 隐式地决定，直接调整 λ 的大小就相当于调整了信赖域半径 Δ 的大小. 因此，我们可构造基于 λ 更新的 LM 方法. 由于 LM 方程(5.7.12)和信赖域子问题的关系由 Fletcher 在 1981 年给出 [45]，因此基于 λ 更新的 LM 方法也被称为 LMF 方法，即每一步求解子问题：

$$\min_{d} \quad \frac{1}{2}\|Jd + r\|_2^2 + \lambda\|d\|_2^2.$$

在 LMF 方法中，设第 k 步产生的迭代方向为 d^k，根据信赖域算法的思想，我们需要计算目标函数的预估下降量和实际下降量的比值 ρ_k，来确定下一步信赖域半径的大小. 这一比值很容易通过公式(5.6.3)计算，其中 $f(x)$ 和 $m_k(d)$ 分别取为(5.7.1)式和(5.7.11)式. 计算 ρ_k 后，我们根据 ρ_k 的大小来更新 λ_k. 当乘子 λ_k 增大时，信赖域半径会变小，反之亦然. 所以 λ_k 的变化策略应该和信赖域算法中 Δ_k 的恰好相反.

有了上面的叙述，接下来就可以给出 LMF 算法的框架了. 通过比较得知 LMF 方法（算法5.10）和信赖域算法5.7的结构非常相似. 算法5.10对 γ_1, γ_2 等参数并不敏感. 但根据信赖域方法的收敛定理5.12，参数 η 可以取大于 0 的值来改善收敛结果.

和 LM 方法相比，LMF 方法每一次迭代只需要求解一次 LM 方程，从而极大地减少了计算量，在编程方面也更容易实现. 所以 LMF 方法在求解最小二乘问题中是很常见的做法.

算法 5.10 LMF 方法

1. 给定初始点 x^0，初始乘子 λ_0，$k \leftarrow 0$.
2. 给定参数 $0 \leqslant \eta < \bar{\rho}_1 < \bar{\rho}_2 < 1$，$\gamma_1 < 1 < \gamma_2$.
3. **while** 未达到收敛准则 **do**
4. 求解 LM 方程 $((J^k)^{\mathrm{T}} J^k + \lambda I)d = -(J^k)^{\mathrm{T}} r^k$ 得到迭代方向 d^k.
5. 根据(5.6.3)式计算下降率 ρ_k.
6. 更新乘子：

$$\lambda_{k+1} = \begin{cases} \gamma_2 \lambda_k, & \rho_k < \bar{\rho}_1, \quad /* \text{ 扩大乘子（缩小信赖域半径）} */ \\ \gamma_1 \lambda_k, & \rho_k > \bar{\rho}_2, \quad /* \text{ 缩小乘子（扩大信赖域半径）} */ \\ \lambda_k, & \text{其他.} \quad /* \text{ 乘子不变 } */ \end{cases}$$

7. 更新自变量：

$$x^{k+1} = \begin{cases} x^k + d^k, & \rho_k > \eta, \quad /* \text{ 只有下降比例足够大才更新 } */ \\ x^k, & \text{其他.} \end{cases}$$

8. $k \leftarrow k + 1$.
9. **end while**

5.7.4 应用举例

相位恢复问题是非线性最小二乘问题的重要应用，它的原始模型为

$$\min_{x \in \mathbb{C}^n} \quad f(x) \xlongequal{\text{def}} \frac{1}{2} \sum_{j=1}^{m} \left(|\bar{a}_j^{\mathrm{T}} x|^2 - b_j \right)^2, \tag{5.7.15}$$

其中 $a_j \in \mathbb{C}^n$ 是已知的采样向量，$b_j \in \mathbb{R}$ 是观测的模长.

该问题的变量为复数，因此我们考虑使用 Wirtinger 导数 [27] 表示其梯度和雅可比矩阵. 对于 $f(x)$，记 $\mathbf{x} = \begin{bmatrix} x \\ \bar{x} \end{bmatrix}$，则

$$\nabla f(x) = \sum_{j=1}^{m} \left(|\bar{a}_j^{\mathrm{T}} x|^2 - b_j \right) a_j \bar{a}_j^{\mathrm{T}} x,$$

以及

$$\nabla f(\mathbf{x}) = \left[\begin{array}{c} \nabla f(x) \\ \hline \overline{\nabla f(x)} \end{array}\right].$$

那么，雅可比矩阵和高斯 – 牛顿矩阵分别为

$$J(\mathbf{x}) = \overline{\left[\begin{array}{cccc} a_1(\bar{a}_1^{\mathrm{T}}x), & a_2(\bar{a}_2^{\mathrm{T}}x), & \cdots, & a_m(\bar{a}_m^{\mathrm{T}}x) \\ \bar{a}_1(a_1^{\mathrm{T}}\bar{x}), & \bar{a}_2(a_2^{\mathrm{T}}\bar{x}), & \cdots, & \bar{a}_m(a_m^{\mathrm{T}}\bar{x}) \end{array}\right]}^{\mathrm{T}},$$

$$\Psi(\mathbf{x}) \overset{\text{def}}{=\!=} \overline{J(\mathbf{x})}^{\mathrm{T}} J(\mathbf{x}) = \sum_{j=1}^{m} \left[\begin{array}{cc} |\bar{a}_j^{\mathrm{T}}x|^2 a_j \bar{a}_j^{\mathrm{T}} & (\bar{a}_j^{\mathrm{T}}x)^2 a_j a_j^{\mathrm{T}} \\ (\overline{\bar{a}_j^{\mathrm{T}}x})^2 \bar{a}_j \bar{a}_j^{\mathrm{T}} & |\bar{a}_j^{\mathrm{T}}x|^2 \bar{a}_j a_j^{\mathrm{T}} \end{array}\right].$$

因此，在第 k 步，高斯 – 牛顿法求解方程

$$\Psi(\mathbf{x}^k) d^k = -\nabla f(\mathbf{x}^k)$$

以得到方向 d^k；LM 方法则求解正则化方程

$$\left(\Psi(\mathbf{x}^k) + \lambda_k\right) d^k = -\nabla f(\mathbf{x}^k), \tag{5.7.16}$$

其中 λ_k 是与 $f(\mathbf{x}^k)$ 相关的参数．这里选取

$$\lambda_k = \begin{cases} 70000n\sqrt{nf(x^k)}, & f(x^k) \geqslant \dfrac{1}{900n}\|x^k\|_2^2, \\ \sqrt{f(x^k)}, & \text{其他}. \end{cases}$$

当 $f(x^k) \geqslant \dfrac{1}{900n}\|x^k\|_2^2$ 时，参数 $\lambda_k = 70000n\sqrt{nf(x^k)}$ 能够保证算法有全局 Q-线性收敛速度．同时，我们利用共轭梯度法求解线性方程(5.7.16)，使得

$$\left\|\left(\Psi(\mathbf{x}^k) + \lambda_k\right) d^k + \nabla f(\mathbf{x}^k)\right\| \leqslant \eta_k \|\nabla f(\mathbf{x}^k)\|,$$

其中 $\eta_k > 0$ 是人为设置的参数.

考虑编码衍射模型，其中信号采集的格式为

$$b_j = \left|\sum_{t=0}^{n-1} x_t \bar{d}_l(t) e^{-i2\pi kt/n}\right|^2, \quad j = (l,k),\ 0 \leqslant k \leqslant n-1,\ 1 \leqslant l \leqslant L.$$

上式表明，对给定的 l，我们采集在波形（waveform）d_l 下信号 $\{x_t\}$ 的衍射图的模长．通过改变 l 和相应的波形 d_l，可以生成一系列编码衍射图．这里假设 $d_l, l = 0,1,\cdots,L$ 是独立同分布的随机向量来模拟实际场景．具体地，

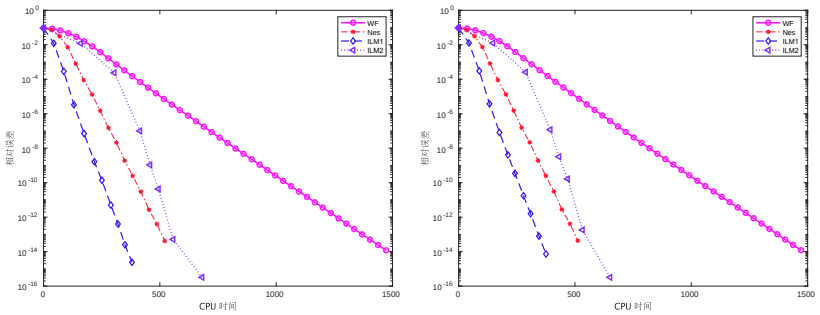
令 $d_l(t)=c_1c_2$，其中

$$c_1=\begin{cases}+1, \text{依概率}0.25,\\ -1, \text{依概率}0.25,\\ +i, \text{依概率}0.25,\\ -i, \text{依概率}0.25,\end{cases}\quad c_2=\begin{cases}\dfrac{\sqrt{2}}{2}, & \text{依概率}0.8,\\ \sqrt{3}, & \text{依概率}0.2.\end{cases}$$

我们分别测试不精确求解正则化方程 (5.7.16)（$\eta_k=0.1$）的 LM 方法（ILM1）以及更精确（$\eta_k=\min\{0.1,\|\nabla f(\mathbf{x}^k)\|\}$）的 LM 方法（ILM2），求解 Wirtinger 梯度下降方法（WF）以及其加速版本 Nesterov 加速算法（Nes）. 真实信号 x 取为两张自然图片，分别为博雅塔和华表的图片，如图 5.16 所示. 这里图片可以看成 $m\times n$ 矩阵，其中行、列指标表示像素点所在位置，取值表示像素点的灰度值. 选取 $L=20$，并收集相应的衍射图模长. 图 5.17 给出了不同算法的收敛情况，其中横坐标为 CPU 时间，纵坐标为当前迭代点 $\mathbf{x}^k$ 与真实信号 x 的相对误差，即 $\min\limits_{\phi\in[0,2\pi]}\dfrac{1}{\|x\|}\|\mathbf{x}^k-xe^{i\phi}\|$.

(a) 博雅塔，图片像素为 601×541　　(b) 华表，图片像素为 601×541

图 5.16　测试图片



(a) 博雅塔 (b) 华表

图 5.17 相对误差和计算时间对比图

5.8 总结

本章简单介绍了若干种经典的无约束优化算法，这些算法的起源都很早，但仍然能够流传至今并经受住时间的考验，正是因为它们在实际问题中取得了很好的效果. 虽然随着技术的进步，更高效、优美的算法不断被发展出来，但是本章所列出的算法形式及其背后所蕴含的思想仍然有很强的指导意义.

常用的精确线搜索算法有黄金分割法（0.618 方法）和斐波那契（Fibnacci）方法，读者可参考 [116] 与 [2]. 有关多项式插值类型的线搜索可以参考 [39]，满足 Wolfe 准则的非精确线搜索算法可参考 [45]. 基于 Wolfe 准则的线搜索算法有标准的子程序，直接调用即可，例如 MINPACK-2 [4]. 除在直线上进行搜索外，我们还有曲线搜索方法，这类方法大多用于牛顿类算法当中，以便产生负曲率方向. 其中基于二阶 Armijo 搜索准则的方法有 Goldfarb 方法 [55] 和 McCormick 方法 [87]；基于二阶 Wolfe 准则的算法有 Moré-Sorensen 方法 [92].

在梯度类算法中我们略去了非线性共轭梯度法的介绍. 早期的共轭梯度法由 Hestenes 和 Stiefel 在 1952 年提出 [68]，主要用于求解正定线性方程组. 它的基本思想是将求解线性方程组转化为一个正定二次函数的最小值问题. 之后 Fletcher 和 Reeves 将其移植到了一般的非线性函数上，并给出了 FR 格式 [44]. 由于线性共轭梯度法有很多等价格式，移植到非线性情况会对应不同的格式，例如 PR 格式 [105] 和 DY 格式 [33]. 非线性共轭梯度法往往比普通梯度法有更好的表现，一些有关收敛性的结果可参考 [34,51,62,140].

牛顿法是利用海瑟矩阵构造下降方向的算法，它有很好的局部收敛性. 但经典牛顿法不够稳定，在实际应用中我们通常使用修正牛顿法求解. 比较古老的修正策略由 Goldstein 在 1967 年提出 [56]，当海瑟矩阵不正定时，选取负梯度作为下降方向. 这一做法完全舍弃了函数的二阶信息，不推荐使用. 而最简单的对海瑟矩阵的修正是 Goldfeld 提出的加常数倍单位矩阵的做法，即使用 $B^k + \tau_k I$ 作为海瑟矩阵，这个策略本质上和信赖域有一定联系. 修正的 Cholesky 分解法是 Gill 和 Murray 提出的 [52]，除此以外还有基于不定矩阵分解 [21] 的 Fletcher-Freeman 算法 [46]，这些算法都能有效提高牛顿法的稳定性. 有关牛顿—共轭梯度法读者可以参考 [60,94]，使用有限差分格式近似牛顿矩阵的方法可参考 [120]，这些算法都能有效减少牛顿法的计算量.

在拟牛顿算法中比较重要的是 BFGS 方法和 L-BFGS 方法. L-BFGS 方法是可以利用在大规模无约束优化问题上的算法，更细致的讨论可以参见 [25,82,94,97]. L-BFGS 算法的代码实现可参考 L-BFGS-B [90,142] 以及其在 GPU 上的一种实现 [43]. 对于有些问题，目标函数的海瑟矩阵是稀疏的，此时如果直接用拟牛顿方法则会破坏稀疏性，所以我们必须根据原问题海瑟矩阵的稀疏结构来设计稀疏的拟牛顿更新，这方面的内容可参考 [107,123]，本书中不做讨论.

在信赖域算法中，除本书介绍的两种算法以外，还有 Powell 提出的 Dogleg 方法 [106]，双折 Dogleg 方法 [38] 以及 Byrd 等人提出的二维子空间法 [26]. 有关截断共轭梯度法的更多性质可参考 [135]. 信赖域算法近几十年的进展总结可参阅 [136].

高斯 – 牛顿法是较常用的求解非线性最小二乘问题的算法. 这个算法充分利用了最小二乘问题的结构，是二阶算法很好的近似. 我们指出，在求解其他问题时也可利用高斯 – 牛顿的思想来构造高效的算法. 例如，若目标函数的海瑟矩阵可自然地写成两部分的和：第一部分容易计算且占主要地位，第二部分难以计算，则可以先精确计算第一部分的值，然后寻找第二部分的近似（或完全舍弃）. 这种做法的效果通常要比直接整体近似海瑟矩阵要好，比较具体的应用可参考 [72].

本章中部分内容编写参考了 [98]，包括线搜索方法、（拟）牛顿类算法、信赖域算法、非线性最小二乘问题算法. 信赖域算法中的截断共轭梯度法参考了 [120]，（次）梯度算法参考了 Lieven Vandenberghe 教授的课件.

习题 6

5.1 设 $f(x)$ 是连续可微函数，d^k 是一个下降方向，且 $f(x)$ 在射线 $\{x^k+\alpha d^k \mid \alpha>0\}$ 上有下界．求证：当 $0<c_1<c_2<1$ 时，总是存在满足 Wolfe 准则(5.1.4a)(5.1.4b)的点．并举一个反例说明当 $0<c_2<c_1<1$ 时，满足 Wolfe 准则的点可能不存在.

5.2 f 为正定二次函数 $f(x)=\frac{1}{2}x^{\mathrm{T}}Ax+b^{\mathrm{T}}x$，$d^k$ 为下降方向，x^k 为当前迭代点．试求出精确线搜索步长

$$\alpha_k=\arg\min_{\alpha>0} f(x^k+\alpha d^k),$$

并由此推出最速下降法的步长满足(5.2.2)式（见定理5.2）.

5.3 利用定理5.5证明推论5.2.

5.4 考虑非光滑函数

$$f(x)=\max_{1\leqslant i\leqslant K} x_i+\frac{1}{2}\|x\|^2,$$

其中 $x\in\mathbb{R}^n$，$K\in[1,n]$ 为一个给定的正整数.

(a) 求出 $f(x)$ 的最小值点 x^* 和对应的函数值 f^*；

(b) 证明 $f(x)$ 在区域 $\{x \mid \|x\|\leqslant R \xlongequal{\text{def}} 1/\sqrt{K}\}$ 上是 G-利普希茨连续的，其中 $G=1+\frac{1}{\sqrt{K}}$；

(c) 设初值 $x^0=0$，考虑使用次梯度算法(5.3.1)对 $\min f(x)$ 进行求解，步长 α_k 可任意选取，证明：存在一种次梯度的取法，在 k（$k<K$）次迭代后，

$$\hat{f}^k-f^*\geqslant\frac{GR}{2(1+\sqrt{K})},$$

其中 $\hat{f}^k$ 的定义和定理 5.5 相同．并根据此例子推出次梯度算法的收敛速度 $\mathcal{O}\left(\frac{GR}{\sqrt{K}}\right)$ 是不能改进的.

5.5 考虑非平方 ℓ_2 正则项优化问题

$$\min\quad f(x)=\frac{1}{2}\|Ax-b\|_2^2+\mu\|x\|_2,$$

其中 $A\in\mathbb{R}^{m\times n}$，注意这个问题并不是岭回归问题.

(a) 若 A 为列正交矩阵，即 $A^{\mathrm{T}}A = I$，利用不可微函数的一阶最优性条件求出该优化问题的显式解；

(b) 对一般的 A 我们可以使用迭代算法来求解这个问题，试设计出不引入次梯度的一种梯度类算法求解该优化问题．提示：$f(x)$ 仅在一点处不可导，若这个点不是最小值点，则次梯度算法和梯度法等价．

5.6 设函数 $f(x) = \|x\|^{\beta}$，其中 $\beta > 0$ 为给定的常数．考虑使用经典牛顿法(5.4.2)对 $f(x)$ 进行极小化，初值 $x^0 \neq 0$．证明：

(a) 若 $\beta > 1$ 且 $\beta \neq 2$，则 x^k 收敛到 0 的速度为 Q-线性的；

(b) 若 $0 < \beta < 1$，则牛顿法发散；

(c) 试解释定理 5.6在 (a) 中不成立的原因．

5.7 设矩阵 A 为 n 阶对称矩阵，d^k 为给定的非零向量．若对任意满足 $\|d\| = \|d^k\|$ 的 $d \in \mathbb{R}^n$，均有 $(d - d^k)^{\mathrm{T}} A (d - d^k) \geqslant 0$，证明：$A$ 是半正定矩阵．

5.8 设 $f(x)$ 为正定二次函数，且假定在迭代过程中 $(s^k - H^k y^k)^{\mathrm{T}} y^k > 0$ 对任意的 k 均满足，其中 H^k 由 SR1 公式(5.5.10)产生的拟牛顿矩阵．证明：

$$H^k y^j = s^j, \quad j = 0, 1, \cdots, k-1,$$

其中 k 是任意给定的整数．这个结论说明对于正定二次函数，SR1 公式产生的拟牛顿矩阵在当前点处满足割线方程，且历史迭代产生的 (s^j, y^j) 也满足割线方程．

5.9 仿照 BFGS 公式的推导过程，试利用待定系数法推导 DFP 公式(5.5.15)．

5.10 (小样本问题) 设 $J(x) \in \mathbb{R}^{m \times n}$ 为最小二乘问题(5.7.1)中 $r(x)$ 在点 x 处的雅可比矩阵，其中 $m \ll n$．设 $J(x)$ 行满秩，证明：

$$\hat{d} = -J(x)^{\mathrm{T}} (J(x) J(x)^{\mathrm{T}})^{-1} r(x)$$

给出了高斯 – 牛顿方程(5.7.3)的一个 ℓ_2 范数最小解．

第六章　约束优化算法

本章考虑约束优化问题

$$\begin{aligned} \min \quad & f(x), \\ \text{s.t.} \quad & x \in \mathcal{X}, \end{aligned} \tag{6.0.1}$$

这里 $\mathcal{X} \subset \mathbb{R}^n$ 为问题的可行域. 与无约束问题不同，约束优化问题中自变量 x 不能任意取值，这导致许多无约束优化算法不能使用. 例如梯度法中沿着负梯度方向下降所得的点未必是可行点，要寻找的最优解处目标函数的梯度也不是零向量. 这使得约束优化问题比无约束优化问题要复杂许多. 本章将介绍一些罚函数法，它们将约束作为惩罚项加到目标函数中，从而转化为我们熟悉的无约束优化问题求解. 此外我们还针对线性规划这一特殊的约束优化问题介绍内点法，它的思想可以被应用到很多一般问题的求解.

6.1　罚函数法

6.1.1　等式约束的二次罚函数法

上一章介绍了各种各样的求解无约束优化问题的方法. 那么，我们能否通过将问题 (6.0.1) 变形为无约束优化问题来求解呢？为此考虑一种简单的情况，假设问题约束中仅含等式约束，即考虑问题

$$\begin{aligned} \min_{x} \quad & f(x), \\ \text{s.t.} \quad & c_i(x) = 0, \quad i \in \mathcal{E}, \end{aligned} \tag{6.1.1}$$

其中变量 $x \in \mathbb{R}^n$，$\mathcal{E}$ 为等式约束的指标集，$c_i(x)$ 为连续函数. 在某些特殊场合下，可以通过直接求解（非线性）方程组 $c_i(x) = 0$ 消去部分变量，将其转化为无约束问题. 但对一般的函数 $c_i(x)$ 来说，变量消去这一操作是不可实现的，我们必须采用其他方法来处理这种问题.

罚函数法的思想是将约束优化问题 (6.1.1) 转化为无约束优化问题来进行求解. 为了保证解的逼近质量，无约束优化问题的目标函数为原约束优化问题的目标函数加上与约束函数有关的惩罚项. 对于可行域外的点，惩罚项为正，即对该点进行惩罚；对于可行域内的点，惩罚项为 0，即不做任何惩罚. 因此，惩罚项会促使无约束优化问题的解落在可行域内.

对于等式约束问题，惩罚项的选取方式有很多，结构最简单的是二次函数. 这里给出二次罚函数的定义.

定义 6.1 (等式约束的二次罚函数) 对等式约束最优化问题 (6.1.1)，定义**二次罚函数**

$$P_E(x,\sigma) = f(x) + \frac{1}{2}\sigma\sum_{i\in\mathcal{E}} c_i^2(x), \tag{6.1.2}$$

其中等式右端第二项称为惩罚项，$\sigma > 0$ 称为罚因子.

由于这种罚函数对不满足约束的点进行惩罚，在迭代过程中点列一般处于可行域之外，因此它也被称为**外点罚函数**. 二次罚函数的特点如下：对于非可行点而言，当 σ 变大时，惩罚项在罚函数中的权重加大，对罚函数求极小，相当于迫使其极小点向可行域靠近；在可行域中，$P_E(x,\sigma)$ 的全局极小点与约束最优化问题 (6.1.1) 的最优解相同.

为了直观理解罚函数的作用，我们给出一个例子.

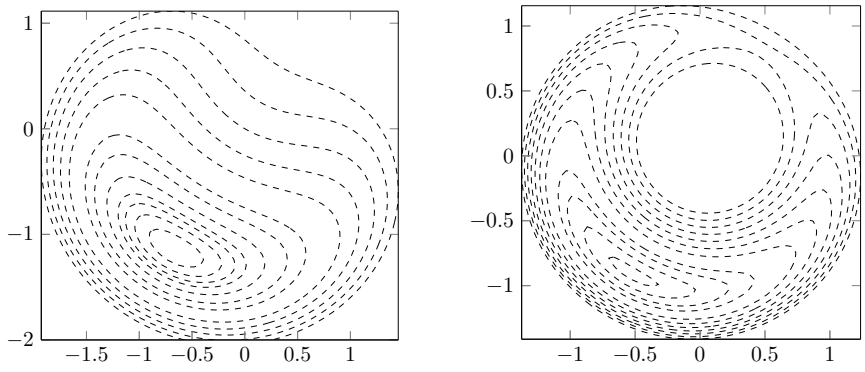
例 6.1 考虑优化问题

$$\begin{aligned} \min \quad & x + \sqrt{3}y, \\ \text{s.t.} \quad & x^2 + y^2 = 1. \end{aligned}$$

容易求出该问题最优解为 $\left(-\frac{1}{2}, -\frac{\sqrt{3}}{2}\right)^{\mathrm{T}}$. 考虑二次罚函数

$$P_E(x,y,\sigma) = x + \sqrt{3}y + \frac{\sigma}{2}(x^2+y^2-1)^2,$$

并在图6.1中绘制出 $\sigma = 1$ 和 $\sigma = 10$ 对应的罚函数的等高线. 可以看出，随着 σ 增大，二次罚函数 $P_E(x,y,\sigma)$ 的最小值和原问题最小值越来越接近，但最优点附近的等高线越来越趋于扁平，这导致求解无约束优化问题的难度变大. 此外，当 $\sigma = 10$ 时函数出现了一个极大值，罚函数图形在 $\left(-\frac{1}{2}, -\frac{\sqrt{3}}{2}\right)^{\mathrm{T}}$ 附近出现了一个鞍点.



(a) $\sigma = 1$ (b) $\sigma = 10$

图 6.1 二次罚函数取不同 σ 时等高线的变化

从以上例子知道，给定罚因子 σ，我们可通过求解 $P_E(x,\sigma)$ 的最小值点作为原问题的近似解. 但实际情况并不总是这样，下面这个例子表明，当 σ 选取过小时罚函数可能无下界.

例 6.2 考虑优化问题

$$\begin{aligned} \min \quad & -x^2 + 2y^2, \\ \text{s.t.} \quad & x = 1. \end{aligned}$$

通过消去变量容易得知最优解就是 $(1,0)^{\mathrm{T}}$. 但考虑罚函数

$$P_E(x,y,\sigma) = -x^2 + 2y^2 + \frac{\sigma}{2}(x-1)^2,$$

对任意的 $\sigma \leqslant 2$，该罚函数是无界的.

出现以上现象的原因是当罚因子过小时，不可行点处的函数下降抵消了罚函数对约束违反的惩罚. 实际上所有外点罚函数法均存在这个问题，因此 σ 的初值选取不应该太小.

我们先在算法6.1中给出等式约束的二次罚函数法，之后对每一步进行具体解释.

算法 6.1 二次罚函数法

1. 给定 $\sigma_1 > 0, x^0, k \leftarrow 1$. 罚因子增长系数 $\rho > 1$.
2. **while** 未达到收敛准则 **do**
3. 以 x^k 为初始点，求解 $x^{k+1} = \arg\min_x P_E(x, \sigma_k)$.
4. 选取 $\sigma_{k+1} = \rho\sigma_k$.
5. $k \leftarrow k + 1$.
6. **end while**

算法6.1的执行过程比较直观：即先选取一系列指数增长的罚因子 σ_k，然后针对每个罚因子求解二次罚函数 $P_E(x, \sigma_k)$ 的最小值点（或局部极小值点）. 这种逐步增加罚因子的做法在实际中被广泛使用，例如在 LASSO 问题求解中这一策略被称为**连续化**（Continuation）. 算法第三行中 $\arg\min$ 的含义是如下情况之一：

(1) x^{k+1} 是罚函数 $P_E(x, \sigma_k)$ 的全局极小解；

(2) x^{k+1} 是罚函数 $P_E(x, \sigma_k)$ 的局部极小解；

(3) x^{k+1} 不是罚函数 $P_E(x, \sigma_k)$ 的严格的极小解，但近似满足一阶最优性条件 $\nabla_x P_E(x^{k+1}, \sigma_k) \approx 0$.

根据前面的叙述，在算法6.1中需要注意如下三点：第一，对参数 σ_k 的选取需要非常小心，若 σ_k 增长太快，则子问题不易求解（具体分析见下一小节末尾对算法数值困难的讨论）. 若增长太慢，则算法需要的外迭代数（算法中的 while 循环）会增多. 一个比较合理的取法是根据当前 $P_E(x, \sigma_k)$ 的求解难度来确定 σ_k 的增幅，若当前子问题收敛很快，可以在下一步选取较大的 σ_{k+1}，否则就不宜过分增大 σ_k. 第二，在前面的例子中我们提到了 $P_E(x, \sigma)$ 在 σ 较小时可能无界，此时迭代就会发散. 当求解子问题时，一旦检测到迭代点发散就应该立即终止迭代并增大罚因子. 第三，子问题求解的精度必须足够精确，为保证收敛，子问题求解误差需要趋于零.

6.1.2 收敛性分析

本小节讨论等式约束的二次罚函数法的收敛性. 为了讨论方便，我们假设对每个 σ_k，$P_E(x, \sigma_k)$ 的最小值点都是存在的. 注意这个假设对某些优化问

题是不对的，其本质原因是因为二次罚函数的惩罚力度不够，因此我们不会使用二次罚函数法去求解不满足此假设的优化问题.

定理 6.1 (二次罚函数法的收敛性 1) 设 x^{k+1} 是 $P_E(x,\sigma_k)$ 的全局极小解，σ_k 单调上升趋于无穷，则 $\{x^k\}$ 的每个极限点 x^* 都是原问题的全局极小解.

证明. 设 $\bar{x}$ 是原问题(6.1.1)的全局极小解，即

$$f(\bar{x}) \leqslant f(x), \quad \forall\, x \text{ 满足 } c_i(x) = 0,\ i \in \mathcal{E}.$$

由定理条件，x^{k+1} 是 $P_E(x,\sigma_k)$ 的全局极小值，我们有 $P_E(x^{k+1},\sigma_k) \leqslant P_E(\bar{x},\sigma_k)$，即

$$f(x^{k+1}) + \frac{\sigma_k}{2}\sum_{i\in\mathcal{E}} c_i^2(x^{k+1}) \leqslant f(\bar{x}) + \frac{\sigma_k}{2}\sum_{i\in\mathcal{E}} c_i^2(\bar{x}) = f(\bar{x}), \tag{6.1.3}$$

整理可得

$$\sum_{i\in\mathcal{E}} c_i^2(x^{k+1}) \leqslant \frac{2}{\sigma_k}(f(\bar{x}) - f(x^{k+1})). \tag{6.1.4}$$

设 x^* 是 $\{x^k\}$ 的一个极限点，为了方便，不妨设 $x^k \to x^*$. 在(6.1.4)式中令 $k \to \infty$，根据 $c_i(x)$ 和 $f(x)$ 的连续性以及 $\sigma_k \to +\infty$ 可知

$$\sum_{i\in\mathcal{E}} c_i^2(x^*) = 0.$$

这说明 x^* 是原问题的一个可行解. 由(6.1.3)式可得 $f(x^{k+1}) \leqslant f(\bar{x})$，两边取极限得 $f(x^*) \leqslant f(\bar{x})$. 由 $\bar{x}$ 的最优性可知 $f(x^*) = f(\bar{x})$，即 x^* 也是全局极小解. □

以上定理表明，若可以找到子问题的全局极小解，则它们的极限点为原问题的最小值点. 但实际应用当中，求 $P_E(x,\sigma_k)$ 的全局极小解是难以做到的，我们只能将子问题求解到一定精度. 因此定理6.1的应用场合十分有限. 下面给出另一个定理，它从最优性条件给出了迭代点列的收敛性.

定理 6.2 (二次罚函数法的收敛性 2) 设 $f(x)$ 与 $c_i(x), i \in \mathcal{E}$ 连续可微，正数序列 $\varepsilon_k \to 0, \sigma_k \to +\infty$，在算法6.1中，子问题的解 x^{k+1} 满足 $\|\nabla_x P_E(x^{k+1},\sigma_k)\| \leqslant \varepsilon_k$，而对 $\{x^k\}$ 的任何极限点 x^*，都有 $\{\nabla c_i(x^*), i \in \mathcal{E}\}$ 线性无关，则 x^* 是等式约束最优化问题(6.1.1) 的 KKT 点，且

$$\lim_{k\to\infty}(-\sigma_k c_i(x^{k+1})) = \lambda_i^*, \quad \forall\, i \in \mathcal{E}, \tag{6.1.5}$$

其中 λ_i^* 是约束 $c_i(x^*) = 0$ 对应的拉格朗日乘子.

证明. 容易求出 $P_E(x,\sigma_k)$ 的梯度为

$$\nabla P_E(x,\sigma_k) = \nabla f(x) + \sum_{i\in\mathcal{E}} \sigma_k c_i(x)\nabla c_i(x). \tag{6.1.6}$$

根据子问题求解的终止准则，对 x^{k+1} 我们有

$$\left\| \nabla f(x^{k+1}) + \sum_{i\in\mathcal{E}} \sigma_k c_i(x^{k+1})\nabla c_i(x^{k+1}) \right\| \leqslant \varepsilon_k. \tag{6.1.7}$$

利用三角不等式 $\|a\| - \|b\| \leqslant \|a+b\|$ 可以把(6.1.7)式变形为

$$\left\| \sum_{i\in\mathcal{E}} c_i(x^{k+1})\nabla c_i(x^{k+1}) \right\| \leqslant \frac{1}{\sigma_k}(\varepsilon_k + \|\nabla f(x^{k+1})\|). \tag{6.1.8}$$

为了方便，不妨设 $\{x^k\}$ 收敛于 x^*，在(6.1.8)式中不等号两边同时令 $k\to\infty$，根据 $f(x), c_i(x)$ 的连续性，

$$\sum_{i\in\mathcal{E}} c_i(x^*)\nabla c_i(x^*) = 0.$$

又由于 $\nabla c_i(x^*)$ 线性无关，此时必有 $c_i(x^*) = 0, \forall\, i\in\mathcal{E}$. 这表明 x^* 实际上是一个可行点.

以下我们说明点 x^* 满足 KKT 条件中的梯度条件，为此需要构造拉格朗日乘子 $\lambda^* = (\lambda_1^*, \lambda_2^*, \cdots, \lambda_{|\mathcal{E}|}^*)^{\mathrm{T}}$，其中 $|\mathcal{E}|$ 表示 $\mathcal{E}$ 中元素的个数. 记

$$\nabla c(x) = [\nabla c_i(x)]_{i\in\mathcal{E}}, \tag{6.1.9}$$

并定义 $\lambda_i^k = -\sigma_k c_i(x^{k+1})$，$\lambda^k = (\lambda_1^*, \lambda_2^*, \cdots, \lambda_{|\mathcal{E}|}^*)^{\mathrm{T}}$，则梯度式 (6.1.6) 可以改写为

$$\nabla c(x^{k+1})\lambda^k = \nabla f(x^{k+1}) - \nabla P_E(x^{k+1}, \sigma_k).$$

由条件知 $\nabla c(x^*)$ 是列满秩矩阵，而 $x^k\to x^*$，因此当 k 充分大时，$\nabla c(x^{k+1})$ 应该是列满秩矩阵，进而可以利用 $\nabla c(x^{k+1})$ 的广义逆来表示 λ^k:

$$\lambda^k = (\nabla c(x^{k+1})^{\mathrm{T}}\nabla c(x^{k+1}))^{-1}\nabla c(x^{k+1})^{\mathrm{T}}(\nabla f(x^{k+1}) - \nabla_x P_E(x^{k+1},\sigma_k)).$$

等式两侧关于 k 取极限，并注意到 $\nabla_x P_E(x^{k+1},\sigma_k)\to 0$，我们有

$$\lambda^* \overset{\text{def}}{=\!=} \lim_{k\to\infty}\lambda^k = (\nabla c(x^*)^{\mathrm{T}}\nabla c(x^*))^{-1}\nabla c(x^*)^{\mathrm{T}}\nabla f(x^*).$$

最后在梯度表达式(6.1.6)中令 $k\to\infty$ 可得

$$\nabla f(x^*) - \nabla c(x^*)\lambda^* = 0.$$

这说明 KKT 条件中的梯度条件成立，λ^* 就是点 x^* 对应的拉格朗日乘子. □

在定理 6.2 的证明过程中，还可以得到一个推论：不管 $\{\nabla c_i(x^*)\}$ 是否线性无关，通过算法6.1给出解 x^k 的聚点总是 $\phi(x)=\|c(x)\|^2$ 的一个稳定点. 这说明即便没有找到可行解，我们也找到了使得约束 $c(x)=0$ 违反度相对较小的一个解. 此外，定理6.2 虽然不要求每一个子问题精确求解，但要获得原问题的解，子问题解的精度需要越来越高. 它并没有给出一个非渐进的误差估计，即没有说明当给定原问题解的目标精度时，子问题的求解精度 ε_k 应该如何选取.

作为等式约束二次罚函数法的总结，最后简要分析一下算法6.1的数值困难. 我们知道，要想得到原问题的解，罚因子 σ_k 必须趋于正无穷. 以下从矩阵条件数的角度说明，当 σ_k 趋于正无穷时，子问题求解难度会显著变大. 考虑罚函数 $P_E(x,\sigma)$ 的海瑟矩阵：

$$\nabla^2_{xx}P_E(x,\sigma)=\nabla^2 f(x)+\sum_{i\in\mathcal{E}}\sigma c_i(x)\nabla^2 c_i(x)+\sigma\nabla c(x)\nabla c(x)^{\mathrm{T}}, \tag{6.1.10}$$

其中 $\nabla c(x)$ 如(6.1.9)式定义. 我们现在考虑当 x 接近最优点时，海瑟矩阵的变化情况. 由定理6.2，在 $x\approx x^*$ 时，应该有 $-\sigma c_i(x)\approx\lambda_i^*$. 根据这一近似，我们可以使用拉格朗日函数 $L(x,\lambda^*)$ 的海瑟矩阵来近似(6.1.10)式等号右边的前两项：

$$\nabla^2_{xx}P_E(x,\sigma)\approx\nabla^2_{xx}L(x,\lambda^*)+\sigma\nabla c(x)\nabla c(x)^{\mathrm{T}}, \tag{6.1.11}$$

其中 $\nabla c(x)\nabla c(x)^{\mathrm{T}}$ 是一个半正定矩阵且奇异，它有 $(n-|\mathcal{E}|)$ 个特征值都是 0. 注意，(6.1.11)式右边包含两个矩阵：一个定值矩阵和一个最大特征值趋于正无穷的奇异矩阵. 从直观上来说，海瑟矩阵 $\nabla^2_{xx}P_E(x,\sigma)$ 的条件数将会越来越大，这意味着子问题的等高线越来越密集，使用梯度类算法求解将会变得非常困难. 若使用牛顿法，则求解牛顿方程本身就是一个非常困难的问题. 因此在实际应用中，我们不可能令罚因子趋于正无穷.

6.1.3 一般约束问题的二次罚函数法

上一小节仅仅考虑了等式约束优化问题，那么对于不等式约束的问题应该如何设计二次罚函数呢？不等式约束优化问题有如下形式：

$$\begin{aligned}\min\quad & f(x),\\ \text{s.t.}\quad & c_i(x)\leqslant 0,\ i\in\mathcal{I}.\end{aligned} \tag{6.1.12}$$

显然，它和等式约束优化问题最大的不同就是允许 $c_i(x)<0$ 发生，而若采用原来的方式定义罚函数为 $\|c(x)\|^2$，它也会惩罚 $c_i(x)<0$ 的可行点，这显

然不是我们需要的. 针对问题(6.1.12), 我们必须对原有二次罚函数进行改造来得到新的二次罚函数, 它应该具有如下特点: 仅仅惩罚 $c_i(x) > 0$ 的那些点, 而对可行点不作惩罚.

定义 6.2 (不等式约束的二次罚函数) 对不等式约束最优化问题 (6.1.12), 定义**二次罚函数**

$$P_I(x,\sigma) = f(x) + \frac{1}{2}\sigma\sum_{i\in\mathcal{I}}\tilde{c}_i^2(x), \tag{6.1.13}$$

其中等式右端第二项称为惩罚项, $\tilde{c}_i(x)$ 的定义为

$$\tilde{c}_i(x) = \max\{c_i(x),0\}, \tag{6.1.14}$$

常数 $\sigma > 0$ 称为罚因子.

注意到函数 $h(t) = (\max\{t,0\})^2$ 关于 t 是可导的, 因此 $P_I(x,\sigma)$ 的梯度也存在, 可以使用梯度类算法来求解子问题. 然而一般来讲 $P_I(x,\sigma)$ 不是二阶可导的, 因此不能直接利用二阶算法 (如牛顿法) 求解子问题, 这也是不等式约束问题二次罚函数的不足之处. 求解不等式约束问题的罚函数法的结构和算法6.1完全相同, 这里略去相关说明.

一般的约束优化问题可能既含等式约束又含不等式约束, 它的形式为

$$\begin{aligned} \min \quad & f(x), \\ \text{s.t.} \quad & c_i(x) = 0,\ i \in \mathcal{E}, \\ & c_i(x) \leqslant 0,\ i \in \mathcal{I}. \end{aligned} \tag{6.1.15}$$

针对这个问题, 我们只需要将两种约束的罚函数相加就能得到一般约束优化问题的二次罚函数.

定义 6.3 (一般约束的二次罚函数) 对一般约束最优化问题 (6.1.15), 定义**二次罚函数**

$$P(x,\sigma) = f(x) + \frac{1}{2}\sigma\Big[\sum_{i\in\mathcal{E}}c_i^2(x) + \sum_{i\in\mathcal{I}}\tilde{c}_i^2(x)\Big], \tag{6.1.16}$$

其中等式右端第二项称为惩罚项, $\tilde{c}_i(x)$ 的定义如(6.1.14)式, 常数 $\sigma > 0$ 称为罚因子.

同样地, 我们可以使用合适的办法来求解罚函数子问题, 在这里不再叙述具体算法.

6.1.4 应用举例

许多优化问题建模都可以看成是应用了罚函数的思想，因此罚函数法也可自然应用到这类问题的求解中.

1. LASSO 问题求解

考虑 LASSO 问题

$$\min_x \quad \frac{1}{2}\|Ax-b\|^2+\mu\|x\|_1,$$

其中 $\mu>0$ 是正则化参数. 我们知道求解 LASSO 问题的最终目标是为了解决如下基追踪 (BP) 问题：

$$\begin{aligned}\min \quad & \|x\|_1,\\ \text{s.t.} \quad & Ax=b,\end{aligned}$$

在这里 $Ax=b$ 是一个欠定方程组. 注意到 BP 问题是一个等式约束的非光滑优化问题，我们使用二次罚函数作用于等式约束 $Ax=b$，可得

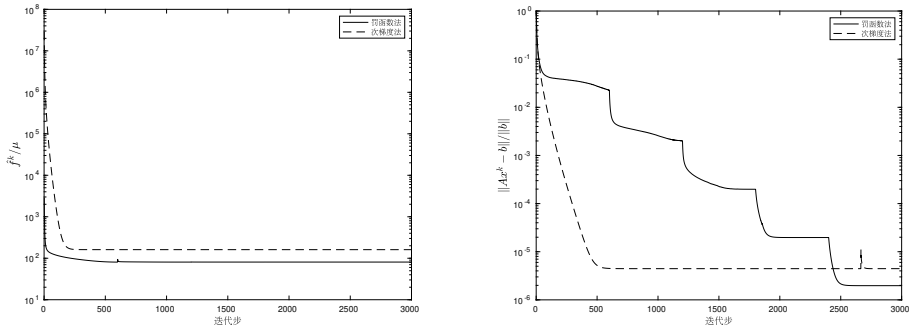
$$\min_x \quad \|x\|_1+\frac{\sigma}{2}\|Ax-b\|^2.$$

令 $\mu=\dfrac{1}{\sigma}$，则容易看出使用 $\dfrac{1}{\mu}$ 作为二次罚因子时，BP 问题的罚函数子问题就等价于 LASSO 问题. 这一观察至少说明了以下两点：第一，LASSO 问题的解和 BP 问题的解本身不等价，当 μ 趋于 0 时，LASSO 问题的解收敛到 BP 问题的解；第二，当 μ 比较小时，根据之前的讨论，此时 BP 问题罚函数比较病态，若直接求解则收敛速度会很慢. 根据罚函数的思想，罚因子应该逐渐增加到无穷，这等价于在 LASSO 问题中先取一个较大的 μ，之后再不断缩小 μ 直至达到我们所要求解的值. 具体算法在算法6.2中给出.

算法 6.2 LASSO 问题求解的罚函数法

1. 给定初值 x^0，最终参数 μ，初始参数 μ_0，因子 $\gamma \in (0,1)$，$k \leftarrow 0$.
2. **while** $\mu_k \geqslant \mu$ **do**
3. 以 x^k 为初值，求解问题 $x^{k+1} = \arg\min_x \left\{ \frac{1}{2}\|Ax-b\|^2 + \mu_k \|x\|_1 \right\}$.
4. **if** $\mu_k = \mu$ **then**
5. 停止迭代，输出 x^{k+1}.
6. **else**
7. 更新罚因子 $\mu_{k+1} = \max\{\mu, \gamma\mu_k\}$.
8. $k \leftarrow k+1$.
9. **end if**
10. **end while**

求解 LASSO 子问题可以使用之前介绍过的次梯度法，也可以使用第七章将提到的多种非光滑函数的优化方法求解. 图6.2展示了分别使用罚函数法



(a) 函数值变化趋势　　(b) 约束违反度变化趋势

图 6.2 使用次梯度法和罚函数法求解 LASSO 问题

和次梯度法求解 LASSO 问题的结果，其中使用与第5.2节中同样的 A 和 b，并取 LASSO 问题的正则化参数为 $\mu = 10^{-3}$. 在罚函数法中，令 μ 从 10 开始，因子 $\gamma = 0.1$，次梯度法选取固定步长 $\alpha = 0.0002$. 从图6.2中我们可明显看到罚函数法的效果比直接使用次梯度法要好，在迭代初期，由于 μ 较大，这意味着约束 $Ax = b$ 可以不满足，从而算法可将优化的重点放到 $\|x\|_1$ 上；随着迭代进行，μ 单调减小，此时算法将更注重于可行性（$\|Ax-b\|$ 的大小）. 直接取 $\mu = 10^{-3}$ 效果不佳，这是因为惩罚项 $\|Ax-b\|^2$ 的权重太大，

次梯度法会尽量使得迭代点满足 $Ax=b$，而忽视了 $\|x\|_1$ 项的作用，图6.2也可说明这一问题. 在每个 LASSO 子问题中我们使用了次梯度法求解，若使用第七章的近似点梯度法求解子问题，那么算法6.2等价于求解 ℓ_1 极小化问题的 FPC (fixed-point continuation) 算法 [63].

6.1.5 其他类型的罚函数法

作为本节的扩展，我们再介绍一些其他类型的罚函数.

1. 内点罚函数法

前面介绍的二次罚函数均属于**外点罚函数**，即在求解过程中允许自变量 x 位于原问题可行域之外，当罚因子趋于无穷时，子问题最优解序列从可行域外部逼近最优解. 自然地，如果我们想要使得子问题最优解序列从可行域内部逼近最优解，则需要构造**内点罚函数**. 顾名思义，内点罚函数在迭代时始终要求自变量 x 不能违反约束，因此它主要用于不等式约束优化问题.

考虑含不等式约束的优化问题(6.1.12)，为了使得迭代点始终在可行域内，当迭代点趋于可行域边界时，我们需要罚函数趋于正无穷，常用的罚函数是**对数罚函数**.

定义 6.4 (对数罚函数) 对不等式约束最优化问题 (6.1.12)，定义**对数罚函数**

$$P_I(x,\sigma)=f(x)-\sigma\sum_{i\in\mathcal{I}}\ln(-c_i(x)), \tag{6.1.17}$$

其中等式右端第二项称为惩罚项，$\sigma>0$ 称为罚因子.

容易看到，$P_I(x,\sigma)$ 的定义域为 $\{x\mid c_i(x)<0\}$，因此在迭代过程中自变量 x 严格位于可行域内部. 当 x 趋于可行域边界时，由于对数罚函数的特点，$P_I(x,\sigma)$ 会趋于正无穷，这说明对数罚函数的极小值严格位于可行域内部. 然而，对原问题(6.1.12)，它的最优解通常位于可行域边界，即 $c_i(x)\leqslant 0$ 中至少有一个取到等号，此时我们需要调整罚因子 σ 使其趋于 0，这会减弱对数罚函数在边界附近的惩罚效果.

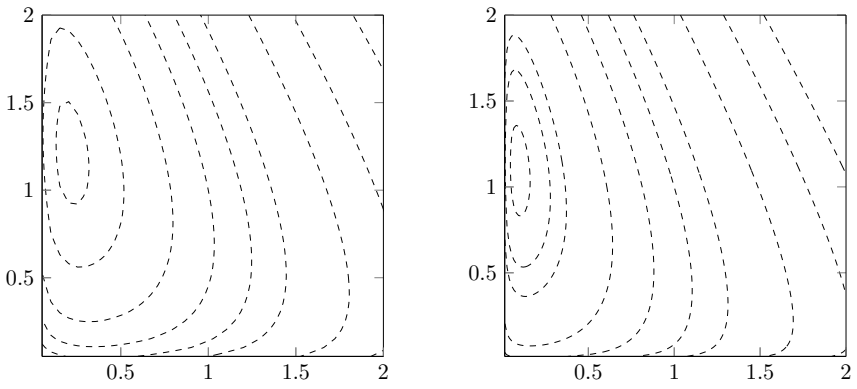
例 6.3 考虑优化问题

$$\begin{aligned}\min\quad & x^2+2xy+y^2+2x-2y,\\ \text{s.t.}\quad & x\geqslant 0,\ y\geqslant 0,\end{aligned}$$

容易求出该问题最优解为 $x=0, y=1$. 我们考虑对数罚函数

$$P_I(x,y,\sigma) = x^2 + 2xy + y^2 + 2x - 2y - \sigma(\ln x + \ln y).$$

并在图6.3中绘制出 $\sigma=1$ 和 $\sigma=0.4$ 对应的等高线. 可以看出, 随着 σ 减小,



(a) $\sigma=1$ (b) $\sigma=0.4$

图 6.3 对数罚函数取不同 σ 时等高线变化

对数罚函数 $P_I(x,y,\sigma)$ 的最小值点和原问题最小值点越来越接近, 但当 x 和 y 趋于可行域边界时, 对数罚函数趋于正无穷.

算法 6.3 给出了基于对数罚函数的优化方法.

算法 6.3 对数罚函数法

1. 给定 $\sigma_0 > 0$, 可行解 x^0, $k \leftarrow 0$. 罚因子缩小系数 $\rho \in (0,1)$.
2. **while** 未达到收敛准则 **do**
3. 以 x^k 为初始点, 求解 $x^{k+1} = \arg\min_x P_I(x,\sigma_k)$.
4. 选取 $\sigma_{k+1} = \rho\sigma_k$.
5. $k \leftarrow k+1$.
6. **end while**

和二次罚函数法不同, 算法 6.3 要求初始点 x^0 是一个可行解, 这是根

据对数罚函数法本身的要求. 常用的收敛准则可以包含

$$\left|\sigma_k \sum_{i\in\mathcal{I}} \ln(-c_i(x^{k+1}))\right| \leqslant \varepsilon,$$

其中 $\varepsilon > 0$ 为给定的精度. 实际上，可以证明（见习题 **6.4**）算法 6.3 产生的迭代点列满足

$$\lim_{k\to\infty} \sigma_k \sum_{i\in\mathcal{I}} \ln(-c_i(x^{k+1})) = 0.$$

同样地，内点罚函数法也会有类似外点罚函数法的数值困难，即当 σ 趋于 0 时，子问题 $P_I(x,\sigma)$ 的海瑟矩阵条件数会趋于无穷，因此对子问题的求解将会越来越困难. 这个现象其实也可从图 6.3 中发现，读者可仿照二次罚函数的情形对对数罚函数进行类似的分析.

2. 精确罚函数法

我们已经介绍了二次罚函数和对数罚函数，它们的一个共同特点就是在求解的时候必须令罚因子趋于正无穷或零，这会带来一定的数值困难. 而对于有些罚函数，在问题求解时不需要令罚因子趋于正无穷（或零），这种罚函数称为**精确罚函数**. 换句话说，若罚因子选取适当，对罚函数进行极小化得到的解恰好就是原问题的精确解. 这个性质在设计算法时非常有用，使用精确罚函数的算法通常会有比较好的性质.

常用的精确罚函数是 ℓ_1 罚函数.

定义 6.5 (ℓ_1 罚函数) 对一般约束最优化问题 (6.1.15)，定义 ℓ_1 **罚函数**

$$P(x,\sigma) = f(x) + \sigma\Big[\sum_{i\in\mathcal{E}} |c_i(x)| + \sum_{i\in\mathcal{I}} \tilde{c}_i(x)\Big] \tag{6.1.18}$$

其中等式右端第二项称为惩罚项，$\tilde{c}_i(x)$ 的定义如(6.1.14)式，常数 $\sigma > 0$ 称为罚因子.

在这里和二次罚函数不同，我们用绝对值代替平方来构造惩罚项，实际上是对约束的 ℓ_1 范数进行惩罚. 注意，ℓ_1 罚函数不是可微函数，求解此罚函数导出的子问题依赖第七章的内容.

下面的定理揭示了 ℓ_1 罚函数的精确性，证明可参考 [64].

定理 6.3 设 x^* 是一般约束优化问题(6.1.15)的一个严格局部极小解，且满足 KKT 条件(4.5.8)，其对应的拉格朗日乘子为 $\lambda_i^*, i \in \mathcal{E}\cup\mathcal{I}$，则当罚因子

$\sigma > \sigma^*$ 时，x^* 也为 $P(x,\sigma)$ 的一个局部极小解，其中

$$\sigma^* = \|\lambda^*\|_\infty \overset{\text{def}}{=\!=} \max_i |\lambda_i^*|.$$

定理6.3说明了对于精确罚函数，当罚因子充分大（不需要是正无穷）时，原问题的极小值点就是 ℓ_1 罚函数的极小值点，这和定理6.1的结果是有区别的.

6.2 增广拉格朗日函数法

在二次罚函数法中，根据定理 6.2，我们有

$$c_i(x^{k+1}) \approx -\frac{\lambda_i^*}{\sigma_k}, \quad \forall i \in \mathcal{E}. \tag{6.2.1}$$

因此，为了保证可行性，罚因子必须趋于正无穷. 此时，子问题因条件数爆炸而难以求解. 那么，是否可以通过对二次罚函数进行某种修正，使得对有限的罚因子，得到的逼近最优解也是可行的？增广拉格朗日函数法就是这样的一个方法.

6.2.1 等式约束优化问题的增广拉格朗日函数法

1. 增广拉格朗日函数法的构造

增广拉格朗日函数法的每一步构造一个增广拉格朗日函数，而该函数的构造依赖于拉格朗日函数和约束的二次罚函数. 具体地，对于等式约束优化问题 (6.1.1)，**增广拉格朗日函数**定义为

$$L_\sigma(x,\lambda) = f(x) + \sum_{i\in\mathcal{E}} \lambda_i c_i(x) + \frac{1}{2}\sigma \sum_{i\in\mathcal{E}} c_i^2(x), \tag{6.2.2}$$

即在拉格朗日函数的基础上，添加约束的二次罚函数. 在第 k 步迭代，给定罚因子 σ_k 和乘子 λ^k，增广拉格朗日函数 $L_{\sigma_k}(x,\lambda^k)$ 的最小值点 x^{k+1} 满足

$$\nabla_x L_{\sigma_k}(x^{k+1},\lambda^k) = \nabla f(x^{k+1}) + \sum_{i\in\mathcal{E}} (\lambda_i^k + \sigma_k c_i(x^{k+1}))\nabla c_i(x^{k+1}) = 0. \tag{6.2.3}$$

对于优化问题 (6.1.1)，其最优解 x^* 以及相应的乘子 λ^* 需满足

$$\nabla f(x^*) + \sum_{i\in\mathcal{E}} \lambda_i^* \nabla c_i(x^*) = 0. \tag{6.2.4}$$

为使增广拉格朗日函数法产生的迭代点列收敛到 x^*，需要保证等式 (6.2.3) (6.2.4) 在最优解处的一致性. 因此，对于充分大的 k，

$$\lambda_i^* \approx \lambda_i^k + \sigma_k c_i(x^{k+1}), \quad \forall i \in \mathcal{E}.$$

上式等价于

$$c_i(x^{k+1}) \approx \frac{1}{\sigma_k}(\lambda_i^* - \lambda_i^k), \tag{6.2.5}$$

所以，当 λ_i^k 足够接近 λ_i^* 时，点 x^{k+1} 处的约束违反度将会远小于 $\frac{1}{\sigma_k}$. 注意，在 (6.2.1)式中约束违反度是正比于 $\frac{1}{\sigma_k}$ 的. 即增广拉格朗日函数法可以通过有效地更新乘子来降低约束违反度. (6.2.5)式表明，乘子的一个有效的更新格式为

$$\lambda_i^{k+1} = \lambda_i^k + \sigma_k c_i(x^{k+1}), \quad \forall i \in \mathcal{E}.$$

那么，我们得到问题(6.1.1)的增广拉格朗日函数法，见算法6.4，其中 $c(x) = [c_i(x)]_{i\in\mathcal{E}}$ 并沿用上一节中的定义

$$\nabla c(x) = [\nabla c_i(x)]_{i\in\mathcal{E}}.$$

算法 6.4 增广拉格朗日函数法

1. 选取初始点 x^0，乘子 λ^0，罚因子 $\sigma_0 > 0$，罚因子更新常数 $\rho > 0$，约束违反度常数 $\varepsilon > 0$ 和精度要求 $\eta_k > 0$. 并令 $k = 0$.
2. **for** $k = 0, 1, 2, \cdots$ **do**
3. 以 x^k 为初始点，求解
$$\min_x \quad L_{\sigma_k}(x, \lambda^k),$$
得到满足精度条件
$$\|\nabla_x L_{\sigma_k}(x, \lambda^k)\| \leqslant \eta_k$$
的解 x^{k+1}.
4. **if** $\|c(x^{k+1})\| \leqslant \varepsilon$ **then**
5. 返回近似解 x^{k+1}, λ^k，终止迭代.
6. **end if**
7. 更新乘子：$\lambda^{k+1} = \lambda^k + \sigma_k c(x^{k+1})$.
8. 更新罚因子：$\sigma_{k+1} = \rho\sigma_k$.
9. **end for**

下面以一个例子来说明增广朗日函数法相较于二次罚函数法在控制约束违反度上的优越性.

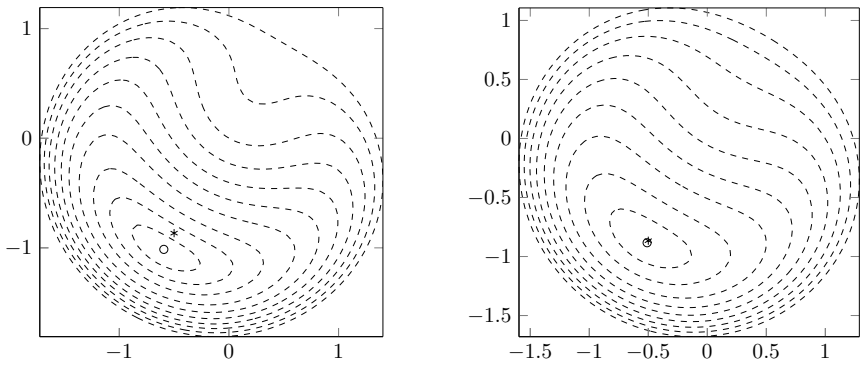
例 6.4 考虑优化问题

$$\begin{aligned} \min \quad & x+\sqrt{3}y, \\ \text{s.t.} \quad & x^2+y^2=1. \end{aligned} \tag{6.2.6}$$

容易求出该问题最优解为 $x^*=\left(-\frac{1}{2},-\frac{\sqrt{3}}{2}\right)^{\mathrm{T}}$，相应的拉格朗日乘子 $\lambda^*=1$. 我们考虑增广拉格朗日函数

$$L_\sigma(x,y,\lambda)=x+\sqrt{3}y+\lambda(x^2+y^2-1)+\frac{\sigma}{2}(x^2+y^2-1)^2,$$

并在图 6.4中绘制出 $L_2(x,y,0.9)$ 的等高线，图中标“$*$”的点为原问题的最优解 x^*，标“$\circ$”的点为罚函数或增广拉格朗日函数的最优解. 对于二次罚函数，其最优解约为 $(-0.5957,-1.0319)$，与最优解 x^* 的欧几里得距离约为 0.1915，约束违反度约为 0.4197. 对于增广拉格朗日函数，其最优解约为 $(-0.5100,-0.8833)$，与最优解 x^* 的欧几里得距离约为 0.02，约束违反度约为 0.0403. 可以看出，增广拉格朗日函数的最优解更接近真实解，与最优解的距离以及约束违反度都约为二次罚函数法的 $\frac{1}{10}$. 需要注意的是，这依赖于乘子的选取.



(a) 二次罚函数　　(b) 增广拉格朗日函数

图 6.4 二次罚函数和增广拉格朗日函数取罚因子 $\sigma=2$ 时等高线变化

随着罚因子 σ_k 的变大，$L_{\sigma_k}(x,\lambda^k)$ 关于 x 的海瑟矩阵的条件数也会越来越大，从而使得迭代点 x^{k+1} 的求解难度提高. 但是，当 σ_k 和 σ_{k+1} 比较接近时，x^k 可以作为求解 x^{k+1} 时的一个初始点，从而加快收敛. 因此，罚因子 σ_k 增长得不能太快. 但如果 σ_k 增长得太慢，迭代点列 $\{x^k\}$ 收敛到原问题解的速度会下降. 在实际中，我们需要注意参数 ρ 的选取，一个经验的取法是 $\rho \in [2,10]$.

2. 收敛性

增广拉格朗日函数作为罚函数的一种，一个关于它的自然的问题是其极小值点和原问题 (6.1.1) 的极小值点有什么关系. 假设 x^*,λ^* 分别为等式约束问题 (6.1.1)的局部极小解和相应的乘子，并且二阶充分条件成立，可以证明，在已知 λ^* 的情况下，对于有限大的 σ，x^* 也为增广拉格朗日函数 $L_\sigma(x,\lambda^*)$ 的严格局部极小解. 当 λ^* 未知时，对于足够接近 λ^* 的 λ 以及足够大的 σ，增广拉格朗日函数 $L_\sigma(x,\lambda)$ 的局部极小解会与 x^* 足够接近. 这也说明了增广拉格朗日函数在一定条件下是精确罚函数.

定理 6.4 *设 x^*,λ^* 分别为问题 (6.1.1)的局部极小解和相应的乘子，并且在点 x^* 处 LICQ 和二阶充分条件成立. 那么，存在一个有限大的常数 $\bar{\sigma}$，使得对任意的 $\sigma \geqslant \bar{\sigma}$，$x^*$ 都是 $L_\sigma(x,\lambda^*)$ 的严格局部极小解. 反之，如果 x^* 为 $L_\sigma(x,\lambda^*)$ 的局部极小解且满足 $c_i(x^*)=0, i\in\mathcal{E}$，那么 x^* 为问题 (6.1.1) 的局部极小解.*

证明. 因为 x^* 为问题 (6.1.1) 的局部极小解且二阶充分条件成立，所以

$$\begin{aligned} &\nabla_x L(x^*,\lambda^*) = \nabla f(x^*) + \sum_{i\in\mathcal{E}} \lambda_i^* \nabla c_i(x^*) = 0, \\ &u^{\mathrm{T}} \nabla_{xx}^2 L(x^*,\lambda^*) u = u^{\mathrm{T}} \left(\nabla^2 f(x^*) + \sum_{i\in\mathcal{E}} \lambda_i^* \nabla^2 c_i(x^*) \right) u \\ &\qquad > 0, \quad \forall u \text{ 满足 } \nabla c(x^*)^{\mathrm{T}} u = 0. \end{aligned} \tag{6.2.7}$$

根据上面的条件，我们来证 x^* 对于 $L_\sigma(x,\lambda^*)$ 的最优性. 因为 $c_i(x^*)=0, i\in\mathcal{E}$，我们有

$$\begin{aligned} &\nabla_x L_\sigma(x^*,\lambda^*) = \nabla_x L(x^*,\lambda^*) = 0, \\ &\nabla_{xx}^2 L_\sigma(x^*,\lambda^*) = \nabla_{xx}^2 L(x^*,\lambda^*) + \sigma \nabla c(x^*) \nabla c(x^*)^{\mathrm{T}}. \end{aligned}$$

对于充分大的 σ，可以证明

$$\nabla_{xx}^2 L_\sigma(x^*,\lambda^*) \succ 0.$$

事实上，如果对于任意的 $\sigma = k$, $k = 1,2,\cdots$，都存在 u_k 满足 $\|u_k\| = 1$，且使得

$$u_k^{\mathrm{T}} \nabla_{xx}^2 L_\sigma(x^*,\lambda^*) u_k = u_k^{\mathrm{T}} \nabla_{xx}^2 L(x^*,\lambda^*) u_k + k\|\nabla c(x^*)^{\mathrm{T}} u_k\|^2 \leqslant 0,$$

则

$$\|\nabla c(x^*)^{\mathrm{T}} u_k\|^2 \leqslant -\frac{1}{k} u_k^{\mathrm{T}} \nabla_{xx}^2 L(x^*,\lambda^*) u_k \to 0, \quad k \to \infty.$$

因为 $\{u_k\}$ 为有界序列，必存在聚点，设为 u. 那么

$$\nabla c(x^*)^{\mathrm{T}} u = 0, \quad u^{\mathrm{T}} \nabla_{xx}^2 L(x^*,\lambda^*) u \leqslant 0,$$

这与 (6.2.7) 式矛盾. 故存在有限大的 $\bar{\sigma}$，使得当 $\sigma \geqslant \bar{\sigma}$ 时，

$$\nabla_{xx}^2 L_\sigma(x^*,\lambda^*) \succ 0,$$

因而 x^* 是 $L_\sigma(x,\lambda^*)$ 的严格局部极小解.

反之，如果 x^* 满足 $c_i(x^*) = 0$ 且为 $L_\sigma(x,\lambda^*)$ 的局部极小解，那么对于任意与 x^* 充分接近的可行点 x，我们有

$$f(x^*) = L_\sigma(x^*,\lambda^*) \leqslant L_\sigma(x,\lambda^*) = f(x).$$

因此，x^* 为问题(6.1.1) 的一个局部极小解. □

对于算法 6.4，通过进一步假设乘子点列的有界性以及收敛点处的约束品性,我们可以证明算法迭代产生的点列 $\{x^k\}$ 会有子列收敛到问题 (6.1.1)的一阶稳定点.

定理 6.5 (增广拉格朗日函数法的收敛性) 假设乘子列 $\{\lambda^k\}$ 是有界的，罚因子 $\sigma_k \to +\infty$, $k \to \infty$，算法 6.4 中精度 $\eta_k \to 0$，迭代点列 $\{x^k\}$ 的一个子序列 x^{k_j+1} 收敛到 x^*，并且在点 x^* 处 LICQ 成立. 那么存在 λ^*，满足

$$\lambda^{k_j+1} \to \lambda^*, \quad j \to \infty,$$
$$\nabla f(x^*) + \nabla c(x^*)\lambda^* = 0, \quad c(x^*) = 0.$$

证明. 对于增广拉格朗日函数 $L_{\sigma_k}(x,\lambda^k)$，

$$\begin{aligned}\nabla_x L_{\sigma_k}(x^{k+1},\lambda^k) &= \nabla f(x^{k+1}) + \nabla c(x^{k+1})(\lambda^k + \sigma_k c(x^{k+1})) \\ &= \nabla f(x^{k+1}) + \nabla c(x^{k+1})\lambda^{k+1} = \nabla_x L(x^{k+1},\lambda^{k+1}).\end{aligned}$$

那么，对于任意使得 $\operatorname{rank}(\nabla c(x^{k_j+1})) = m = |\mathcal{E}|$ 的 k_j（根据假设和点 x^* 处 LICQ 成立，当 x^{k_j+1} 充分接近 x^* 时此式成立），

$$\lambda^{k_j+1} = \left(\nabla c(x^{k_j+1})^{\mathrm{T}} \nabla c(x^{k_j+1})\right)^{-1} \nabla c(x^{k_j+1})^{\mathrm{T}} \left(\nabla_x L_{\sigma_k}(x^{k_j+1}, \lambda^{k_j}) - \nabla f(x^{k_j+1})\right).$$

因为 $\|\nabla_x L_{\sigma_k}(x^{k_j+1}, \lambda^{k_j})\| \leqslant \eta_{k_j} \to 0$，我们有

$$\lambda^{k_j+1} \to \lambda^* \xlongequal{\text{def}} -\left(\nabla c(x^*)^{\mathrm{T}} \nabla c(x^*)\right)^{-1} \nabla c(x^*)^{\mathrm{T}} \nabla f(x^*)$$

以及

$$\nabla_x L(x^*, \lambda^*) = 0.$$

而 $\{\lambda^k\}$ 是有界的，并且 $\lambda^{k_j} + \sigma_{k_j} c(x^{k_j+1}) \to \lambda^*$，所以

$$\{\sigma_{k_j} c(x^{k_j+1})\} \text{ 是有界的.}$$

又 $\sigma_k \to +\infty$，则

$$c(x^*) = 0. \qquad \square$$

定理6.5依赖于乘子列 $\{\lambda_k\}$ 的有界性，$\{x^k\}$ 的子序列收敛性，以及收敛点处的 LICQ. 这里，我们不加证明地给出更一般性的收敛结果，证明过程可以参考 [13]命题 2.7.

定理 6.6 (增广拉格朗日函数法的收敛性——更弱的假设) 假设 x^*, λ^* 分别是问题 (6.1.1)的严格局部极小解和相应的拉格朗日乘子，那么，存在足够大的常数 $\bar{\sigma} > 0$ 和足够小的常数 $\delta > 0$，如果对某个 k，有

$$\frac{1}{\sigma_k}\|\lambda^k - \lambda^*\| < \delta, \quad \sigma_k \geqslant \bar{\sigma},$$

则

$$\lambda^k \to \lambda^*, \quad x^k \to x^*.$$

同时，如果 $\limsup_{k\to\infty} \sigma_k < +\infty$ 且 $\lambda^k \neq \lambda^*, \forall k$，则 $\{\lambda^k\}$ 收敛的速度是 Q-线性的；如果 $\limsup_{k\to\infty} \sigma_k = +\infty$ 且 $\lambda^k \neq \lambda^*, \forall k$，则 $\{\lambda^k\}$ 收敛的速度是 Q-超线性的.

定理6.6不需要假设 $\{\sigma_k\}$ 趋于正无穷（尽管由 $\limsup_{k\to\infty} \sigma_k = +\infty$ 可以推出 Q-超线性收敛）以及 $\{x^k\}$ 的子序列收敛性. 相应地，这里需要找到合适的 λ^k 和 σ_k.

6.2.2 一般约束优化问题的增广拉格朗日函数法

对于一般约束优化问题

$$\begin{aligned} \min \quad & f(x), \\ \text{s.t.} \quad & c_i(x) = 0,\ i \in \mathcal{E}, \\ & c_i(x) \leqslant 0,\ i \in \mathcal{I}, \end{aligned} \tag{6.2.8}$$

也可以定义其增广拉格朗日函数以及设计相应的增广拉格朗日函数法. 在拉格朗日函数的定义中，往往倾向于将简单的约束（比如非负约束、盒子约束等）保留，对复杂的约束引入乘子. 这里，对于带不等式约束的优化问题，我们先通过引入松弛变量将不等式约束转化为等式约束和简单的非负约束，再对保留非负约束形式的拉格朗日函数添加等式约束的二次罚函数来构造增广拉格朗日函数.

1. 增广拉格朗日函数

对于问题 (6.2.8)，通过引入松弛变量可以得到如下等价形式：

$$\begin{aligned} \min_{x,s} \quad & f(x), \\ \text{s.t.} \quad & c_i(x) = 0,\ i \in \mathcal{E}, \\ & c_i(x) + s_i = 0,\ i \in \mathcal{I}, \\ & s_i \geqslant 0,\ i \in \mathcal{I}. \end{aligned} \tag{6.2.9}$$

保留非负约束，可以构造拉格朗日函数

$$L(x,s,\lambda,\mu) = f(x) + \sum_{i\in\mathcal{E}} \lambda_i c_i(x) + \sum_{i\in\mathcal{I}} \mu_i (c_i(x) + s_i),\ s_i \geqslant 0, i \in \mathcal{I}.$$

记问题(6.2.9)中等式约束的二次罚函数为 $p(x,s)$，则

$$p(x,s) = \sum_{i\in\mathcal{E}} c_i^2(x) + \sum_{i\in\mathcal{I}} (c_i(x) + s_i)^2.$$

我们构造增广拉格朗日函数如下：

$$\begin{aligned} L_\sigma(x,s,\lambda,\mu) =& f(x) + \sum_{i\in\mathcal{E}} \lambda_i c_i(x) + \sum_{i\in\mathcal{I}} \mu_i (c_i(x) + s_i) + \\ & \frac{\sigma}{2} p(x,s), \quad s_i \geqslant 0, i \in \mathcal{I}, \end{aligned}$$

其中 σ 为罚因子.

2. 增广拉格朗日函数法

在第 k 步迭代中，给定乘子 λ^k,μ^k 和罚因子 σ_k，我们需要求解如下问题：

$$\min_{x,s} \quad L_{\sigma_k}(x,s,\lambda^k,\mu^k), \quad \text{s.t.} \quad s \geqslant 0 \tag{6.2.10}$$

以得到 x^{k+1},s^{k+1}. 求解问题(6.2.10)的一个有效的方法是投影梯度法（将在第七章的近似点梯度法中介绍）. 另外一种方法是消去 s，求解只关于 x 的优化问题. 具体地，固定 x，关于 s 的子问题可以表示为

$$\min_{s\geqslant 0} \quad \sum_{i\in\mathcal{I}}\mu_i(c_i(x)+s_i)+\frac{\sigma_k}{2}\sum_{i\in\mathcal{I}}(c_i(x)+s_i)^2.$$

根据凸优化问题的最优性理论，s 为以上问题的一个全局最优解，当且仅当

$$s_i=\max\left\{-\frac{\mu_i}{\sigma_k}-c_i(x),0\right\}, \quad i\in\mathcal{I}. \tag{6.2.11}$$

将 s_i 的表达式代入 L_{σ_k} 我们有

$$\begin{aligned} L_{\sigma_k}(x,\lambda^k,\mu^k) =& f(x)+\sum_{i\in\mathcal{E}}\lambda_i c_i(x)+\frac{\sigma_k}{2}\sum_{i\in\mathcal{E}}c_i^2(x)+ \\ & \frac{\sigma_k}{2}\sum_{i\in\mathcal{I}}\left(\max\left\{\frac{\mu_i}{\sigma_k}+c_i(x),0\right\}^2-\frac{\mu_i^2}{\sigma_k^2}\right), \end{aligned} \tag{6.2.12}$$

其为关于 x 的连续可微函数（如果 $f(x),c_i(x),i\in\mathcal{I}\cup\mathcal{E}$ 连续可微）. 因此，问题 (6.2.10)等价于

$$\min_{x\in\mathbb{R}^n} \quad L_{\sigma_k}(x,\lambda^k,\mu^k),$$

并可以利用梯度法进行求解. 这样做的一个好处是，我们消去了变量 s，从而在低维空间 $\mathbb{R}^n$ 中（问题 (6.2.10)的决策空间为 $\mathbb{R}^{n+|\mathcal{I}|}$）求解极小点.

对于问题 (6.2.9)，其最优解 x^*,s^* 和乘子 λ^*,μ^* 需满足 KKT 条件：

$$\begin{aligned} &0=\nabla f(x^*)+\sum_{i\in\mathcal{E}}\lambda_i^*\nabla c_i(x^*)+\sum_{i\in\mathcal{I}}\mu_i^*\nabla c_i(x^*), \\ &\mu_i^*\geqslant 0,\ i\in\mathcal{I}, \\ &s_i^*\geqslant 0,\ i\in\mathcal{I}. \end{aligned}$$

问题 (6.2.10)的最优解 x^{k+1},s^{k+1} 满足

$$\begin{aligned} 0=&\nabla f(x^{k+1})+\sum_{i\in\mathcal{E}}\left(\lambda_i^k+\sigma_k c_i(x^{k+1})\right)\nabla c_i(x^{k+1})+ \\ &\sum_{i\in\mathcal{I}}\left(\mu_i^k+\sigma_k(c_i(x^{k+1})+s_i^{k+1})\right)\nabla c_i(x^{k+1}), \\ s_i^{k+1}=&\max\left\{-\frac{\mu_i^k}{\sigma_k}-c_i(x^{k+1}),0\right\}, \quad i\in\mathcal{I}. \end{aligned}$$

对比问题(6.2.9)和问题(6.2.10)的 KKT 条件，易知乘子的更新格式为

$$\lambda_i^{k+1} = \lambda_i^k + \sigma_k c_i(x^{k+1}), \quad i \in \mathcal{E},$$
$$\mu_i^{k+1} = \max\{\mu_i^k + \sigma_k c_i(x^{k+1}), 0\}, \quad i \in \mathcal{I}.$$

对于等式约束，约束违反度定义为

$$v_k(x^{k+1}) = \sqrt{\sum_{i\in\mathcal{E}} c_i^2(x^{k+1}) + \sum_{i\in\mathcal{I}} \left(c_i(x^{k+1}) + s_i^{k+1}\right)^2}.$$

根据 (6.2.11) 式消去 s，约束违反度为

$$v_k(x^{k+1}) = \sqrt{\sum_{i\in\mathcal{E}} c_i^2(x^{k+1}) + \sum_{i\in\mathcal{I}} \max\left\{c_i(x^{k+1}), -\frac{\mu_i^k}{\sigma_k}\right\}^2}.$$

综上，我们给出约束优化问题 (6.2.10) 的增广拉格朗日函数法，见算法 6.5. 该算法和算法 6.4 结构相似，但它给出了算法参数的一种具体更新方式. 每次计算出子问题的近似解 x^{k+1} 后，算法需要判断约束违反度 $v_k(x^{k+1})$ 是否满足精度要求. 若满足，则进行乘子的更新，并提高子问题求解精度，此时罚因子不变；若不满足，则不进行乘子的更新，并适当增大罚因子以便得到约束违反度更小的解.

6.2.3 凸优化问题的增广拉格朗日函数法

考虑凸优化问题：

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x), \\ \text{s.t.} \quad & c_i(x) \leqslant 0,\ i = 1,2,\cdots,m, \end{aligned} \tag{6.2.13}$$

其中 $f : \mathbb{R}^n \to \mathbb{R}, c_i : \mathbb{R}^n \to \mathbb{R}, i = 1,2,\cdots,m$ 为闭凸函数. 为了叙述的方便，这里考虑不等式形式的凸优化问题. 定义可行域为 $\mathcal{X} = \{x \mid c_i(x) \leqslant 0,\ i = 1,2,\cdots,m\}$.

对于问题 (6.2.13)，根据上一小节介绍的不等式约束的增广拉格朗日函数表达式 (6.2.12)（这里 $\mathcal{E} = \emptyset$），其增广拉格朗日函数为

$$L_\sigma(x,\lambda) = f(x) + \frac{\sigma}{2}\sum_{i=1}^m \left(\max\left\{\frac{\lambda_i}{\sigma} + c_i(x), 0\right\}^2 - \frac{\lambda_i^2}{\sigma^2}\right),$$

其中 λ 和 σ 分别为乘子以及罚因子.

算法 6.5 问题(6.2.10)的增广拉格朗日函数法

1. 选取初始点 x^0，乘子 λ^0, μ^0，罚因子 $\sigma_0 > 0$，约束违反度常数 $\varepsilon > 0$，精度常数 $\eta > 0$，以及常数 $0 < \alpha \leqslant \beta \leqslant 1$ 和 $\rho > 1$. 令 $\eta_0 = \dfrac{1}{\sigma_0}$, $\varepsilon_0 = \dfrac{1}{\sigma_0^{\alpha}}$ 以及 $k = 0$.
2. **for** $k = 0, 1, 2, \cdots$ **do**
3. 以 x^k 为初始点，求解

$$\min_x \quad L_{\sigma_k}(x, \lambda^k, \mu^k),$$

得到满足精度条件

$$\|L_{\sigma_k}(x^{k+1}, \lambda^k, \mu^k)\|_2 \leqslant \eta_k$$

的解 x^{k+1}.

4. **if** $v_k(x^{k+1}) \leqslant \varepsilon_k$ **then**
5. **if** $v_k(x^{k+1}) \leqslant \varepsilon$ 且 $\|\nabla_x L_{\sigma_k}(x^{k+1}, \lambda^k, \mu^k)\|_2 \leqslant \eta$ **then**
6. 得到逼近解 $x^{k+1}, \lambda^k, \mu^k$，终止迭代.
7. **end if**
8. 更新乘子：

$$\begin{aligned} \lambda_i^{k+1} &= \lambda_i^k + \sigma_k c_i(x^{k+1}), \quad i \in \mathcal{E}, \\ \mu_i^{k+1} &= \max\{\mu_i^k + \sigma_k c_i(x^{k+1}), 0\}, \quad i \in \mathcal{I}. \end{aligned}$$

9. 罚因子不变：$\sigma_{k+1} = \sigma_k$.
10. 减小子问题求解误差和约束违反度：$\eta_{k+1} = \dfrac{\eta_k}{\sigma_{k+1}}$, $\varepsilon_{k+1} = \dfrac{\varepsilon_k}{\sigma_{k+1}^{\beta}}$.
11. **else**
12. 乘子不变：$\lambda^{k+1} = \lambda^k$.
13. 更新罚因子：$\sigma_{k+1} = \rho\sigma_k$.
14. 调整子问题求解误差和约束违反度：$\eta_{k+1} = \dfrac{1}{\sigma_{k+1}}$, $\varepsilon_{k+1} = \dfrac{1}{\sigma_{k+1}^{\alpha}}$.
15. **end if**
16. **end for**

给定一列单调递增的乘子 $\sigma_k \uparrow \sigma_\infty$，以及初始乘子 λ^0，问题 (6.2.13) 的增广拉格朗日函数法为

$$\begin{cases} x^{k+1} \approx \arg\min\limits_{x\in\mathbb{R}^n} L_{\sigma_k}(x,\lambda^k), \\ \lambda^{k+1} = \lambda^k + \sigma_k \nabla_\lambda L_{\sigma_k}(x^{k+1},\lambda^k) = \max\{0, \lambda^k + \sigma_k c(x^{k+1})\}. \end{cases} \tag{6.2.14}$$

为了方便叙述，以下定义 $\phi_k(x) = L_{\sigma_k}(x,\lambda^k)$. 由于 $\phi_k(x)$ 的最小值点的显式表达式通常是未知的，我们往往调用迭代算法求其一个近似解. 为了保证收敛性，我们要求该近似解至少满足如下非精确条件之一（参考 [111,133,141]）：

$$\phi_k(x^{k+1}) - \inf\phi_k \leqslant \frac{\varepsilon_k^2}{2\sigma_k}, \quad \varepsilon_k \geqslant 0, \sum_{k=1}^{\infty}\varepsilon_k < +\infty, \tag{6.2.15}$$

$$\phi_k(x^{k+1}) - \inf\phi_k \leqslant \frac{\delta_k^2}{2\sigma_k}\|\lambda^{k+1}-\lambda^k\|_2^2, \quad \delta_k \geqslant 0, \sum_{k=1}^{\infty}\delta_k < +\infty, \tag{6.2.16}$$

$$\operatorname{dist}(0,\partial\phi_k(x^{k+1})) \leqslant \frac{\delta_k'}{\sigma_k}\|\lambda^{k+1}-\lambda^k\|_2, \quad 0 \leqslant \delta_k' \to 0, \tag{6.2.17}$$

其中 $\varepsilon_k, \delta_k, \delta_k'$ 是人为设定的参数，$\operatorname{dist}(0,\partial\phi_k(x^{k+1}))$ 表示 0 到集合 $\partial\phi_k(x^{k+1})$ 的欧几里得距离. 根据 λ^{k+1} 的更新格式 (6.2.14)，容易得知

$$\begin{aligned} \|\lambda^{k+1}-\lambda^k\|_2 &= \|\max\{0,\lambda^k+\sigma_k c(x^{k+1})\} - \lambda^k\|_2 \\ &= \|\max\{-\lambda^k, \sigma_k c(x^{k+1})\}\|_2. \end{aligned}$$

由于 $\inf\phi_k$ 是未知的，直接验证上述不精确条件中的 (6.2.15) 式和 (6.2.16) 式是数值上不可行的. 但是，如果 ϕ_k 是 α-强凸函数（在某些应用中可以计算出 α 或得到其估计值），那么（见习题 **2.10**）

$$\phi_k(x) - \inf\phi_k \leqslant \frac{1}{2\alpha}\operatorname{dist}^2(0,\partial\phi_k(x)). \tag{6.2.18}$$

根据 (6.2.18)式，可以进一步构造如下数值可验证的不精确条件：

$$\operatorname{dist}(0,\partial\phi_k(x^{k+1})) \leqslant \sqrt{\frac{\alpha}{\sigma_k}}\varepsilon_k, \quad \varepsilon_k \geqslant 0, \sum_{k=1}^{\infty}\varepsilon_k < +\infty,$$

$$\operatorname{dist}(0,\partial\phi_k(x^{k+1})) \leqslant \sqrt{\frac{\alpha}{\sigma_k}}\delta_k\|\lambda^{k+1}-\lambda^k\|_2, \quad \delta_k \geqslant 0, \sum_{k=1}^{\infty}\delta_k < +\infty,$$

$$\operatorname{dist}(0,\partial\phi_k(x^{k+1})) \leqslant \frac{\delta_k'}{\sigma_k}\|\lambda^{k+1}-\lambda^k\|_2, \quad 0 \leqslant \delta_k' \to 0.$$

这里，我们给出不精确条件 (6.2.15)下的增广拉格朗日函数法 (6.2.14)的收敛性. 证明细节可以参考 [111]定理 4.

定理 6.7 (凸问题的增广拉格朗日函数法的收敛性) 假设 $\{x^k\},\{\lambda^k\}$ 为问题 (6.2.13)的增广拉格朗日函数法(6.2.14) 生成的序列，x^{k+1} 满足不精确条件 (6.2.15). 如果问题(6.2.13)的 Slater 约束品性成立，那么序列 $\{\lambda^k\}$ 是有界且收敛的，记极限为 λ^∞，则 λ^∞ 为对偶问题的一个最优解.

如果存在一个 γ，使得下水平集 $\{x\in\mathcal{X}|f(x)\leqslant\gamma\}$ 是非空有界的，那么序列 $\{x^k\}$ 也是有界的，并且其所有的聚点都是问题 (6.2.13)的最优解.

注 6.1 这里的乘子 λ^k 与文章 [111] 中的互为相反数，其原因是在构造拉格朗日函数的时候，我们引入的乘子为 $-\lambda(\geqslant 0)$，而文章 [111] 引入的乘子为 $\lambda(\geqslant 0)$.

注 6.2 和定理 6.7 类似，同样有基于不精确条件 (6.2.16) 和 (6.2.17) 的收敛性结果. 见 [111][定理 5].

6.2.4 基追踪问题的增广拉格朗日函数法

这一小节将以基追踪（BP）问题为例讨论增广拉格朗日函数法. 本小节的内容主要参考了 [134,141].

设 $A\in\mathbb{R}^{m\times n}(m\leqslant n), b\in\mathbb{R}^m, x\in\mathbb{R}^n$，BP 问题 (3.1.4) 为

$$\min_{x\in\mathbb{R}^n} \quad \|x\|_1, \quad \text{s.t.} \quad Ax=b. \tag{6.2.19}$$

引入拉格朗日乘子 $y\in\mathbb{R}^m$，BP 问题 (6.2.19) 的拉格朗日函数为

$$L(x,y)=\|x\|_1+y^{\mathrm{T}}(Ax-b),$$

那么对偶函数

$$g(y)=\inf_x L(x,y)=\begin{cases}-b^{\mathrm{T}}y, & \|A^{\mathrm{T}}y\|_\infty\leqslant 1,\\ -\infty, & \text{其他}.\end{cases}$$

因此，我们得到如下对偶问题：

$$\min_{y\in\mathbb{R}^m} \quad b^{\mathrm{T}}y, \quad \text{s.t.} \quad \|A^{\mathrm{T}}y\|_\infty\leqslant 1. \tag{6.2.20}$$

通过引入变量 s，上述问题可以等价地写成

$$\min_{y\in\mathbb{R}^m, s\in\mathbb{R}^n} \quad b^{\mathrm{T}}y, \quad \text{s.t.} \quad A^{\mathrm{T}}y-s=0,\ \|s\|_\infty\leqslant 1. \tag{6.2.21}$$

下面讨论如何对原始问题和对偶问题应用增广拉格朗日函数法.

1. 原始问题的增广拉格朗日函数法

引入罚因子 σ 和乘子 λ，问题 (6.2.19)的增广拉格朗日函数为

$$L_\sigma(x,\lambda) = \|x\|_1 + \lambda^{\mathrm{T}}(Ax-b) + \frac{\sigma}{2}\|Ax-b\|_2^2. \tag{6.2.22}$$

在增广拉格朗日函数法的一般理论中，需要罚因子 σ 足够大来保证迭代收敛（控制约束违反度）. 对于 BP 问题(6.2.19)，后面可以证明，对于固定的非负罚因子也能够保证收敛性（尽管在实际中动态调整罚因子可能会使得算法更快收敛）. 现在考虑固定罚因子 σ 情形的增广拉格朗日函数法. 在第 k 步迭代，更新格式为

$$\begin{cases} x^{k+1} = \arg\min\limits_{x\in\mathbb{R}^n} L_\sigma(x,\lambda^k) = \arg\min\limits_{x\in\mathbb{R}^n} \left\{ \|x\|_1 + \dfrac{\sigma}{2}\|Ax-b+\dfrac{\lambda^k}{\sigma}\|_2^2 \right\}, \\ \lambda^{k+1} = \lambda^k + \sigma(Ax^{k+1}-b). \end{cases} \tag{6.2.23}$$

设迭代的初始点为 $x^0=\lambda^0=0$. 考虑迭代格式(6.2.23)中的第一步，假设 x^{k+1} 为 $L_\sigma(x,\lambda^k)$ 的一个全局极小解，那么

$$0 \in \partial\|x^{k+1}\|_1 + \sigma A^{\mathrm{T}}\left(Ax^{k+1}-b+\frac{\lambda^k}{\sigma}\right).$$

因此，

$$-A^{\mathrm{T}}\lambda^{k+1} \in \partial\|x^{k+1}\|_1. \tag{6.2.24}$$

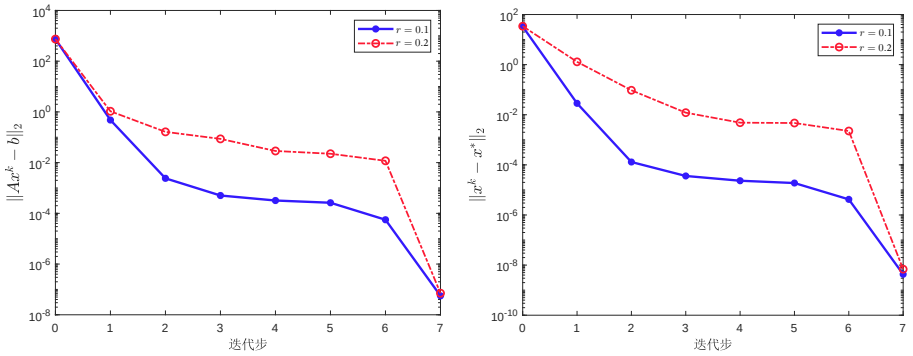
满足上式的 x^{k+1} 往往是不能显式得到的，需要采用迭代算法来进行求解，比如上一章介绍的次梯度法，以及第七章将介绍的近似点梯度法，等等.

这里沿用第5.2节中的 A 和 b 的生成方式，且选取不同的稀疏度 $r=0.1$ 和 $r=0.2$. 我们固定罚因子 σ，并采用近似点梯度法作为求解器，不精确地求解关于 x 的子问题以得到 x^{k+1}. 具体地，设置求解精度 $\eta_k = 10^{-k}$，并且使用 BB 步长作为线搜索初始步长. 图6.5展示了算法产生的迭代点与最优点的距离变化以及约束违反度的走势. 从图6.5中可以看到：对于 BP 问题，固定的 σ 也可以保证增广拉格朗日函数法收敛.

2. 对偶问题的增广拉格朗日函数法

考虑对偶问题(6.2.21)：

$$\min_{y\in\mathbb{R}^m, s\in\mathbb{R}^n} \quad b^{\mathrm{T}}y, \quad \text{s.t.} \quad A^{\mathrm{T}}y - s = 0, \quad \|s\|_\infty \leqslant 1.$$



(a) 约束违反度 (b) 与最优点的距离

图 6.5 增广拉格朗日函数法求解 BP 问题

引入拉格朗日乘子 λ 和罚因子 σ，增广拉格朗日函数为

$$L_\sigma(y,s,\lambda) = b^{\mathrm{T}}y + \lambda^{\mathrm{T}}(A^{\mathrm{T}}y - s) + \frac{\sigma}{2}\|A^{\mathrm{T}}y - s\|_2^2, \quad \|s\|_\infty \leqslant 1.$$

那么，增广拉格朗日函数法的迭代格式为：

$$\begin{cases} (y^{k+1}, s^{k+1}) = \arg\min\limits_{y,\|s\|_\infty \leqslant 1} L_{\sigma_k}(y,s,\lambda^k) \\ \qquad\qquad = \arg\min\limits_{y,\|s\|_\infty \leqslant 1} \left\{ b^{\mathrm{T}}y + \dfrac{\sigma_k}{2}\|A^{\mathrm{T}}y - s + \dfrac{\lambda^k}{\sigma_k}\|_2^2 \right\}, \\ \lambda^{k+1} = \lambda^k + \sigma_k(A^{\mathrm{T}}y^{k+1} - s^{k+1}), \\ \sigma_{k+1} = \min\{\rho\sigma_k, \bar{\sigma}\}, \end{cases}$$

其中 $\rho > 1$ 和 $\bar{\sigma} < +\infty$ 为算法参数. 由于 (y^{k+1}, s^{k+1}) 的显式表达式是未知的，我们需要利用迭代算法来进行求解.

除了利用投影梯度法求解关于 (y,s) 的联合最小化问题外，还可以利用最优性条件将 s 用 y 来表示，转而求解只关于 y 的最小化问题. 具体地，关于 s 的极小化问题为

$$\min_s \quad \frac{\sigma}{2}\|A^{\mathrm{T}}y - s + \frac{\lambda}{\sigma}\|_2^2, \quad \text{s.t.} \quad \|s\|_\infty \leqslant 1.$$

通过简单地推导，可知

$$s = \mathcal{P}_{\|s\|_\infty \leqslant 1}\left(A^{\mathrm{T}}y + \frac{\lambda}{\sigma}\right), \tag{6.2.25}$$

其中 $\mathcal{P}_{\|s\|_\infty\leqslant 1}$ 为集合 $\{s:\|s\|_\infty\leqslant 1\}$ 的投影算子，即

$$\mathcal{P}_{\|s\|_\infty\leqslant 1}(x)=\max\{\min\{x,1\},-1\}.$$

将 s 的表达式代入增广拉格朗日函数中，我们得到

$$L_\sigma(y,\lambda)=b^{\mathrm{T}}y+\frac{\sigma}{2}\left\|\psi\left(A^{\mathrm{T}}y+\frac{\lambda}{\sigma}\right)\right\|_2^2-\frac{\lambda^2}{2\sigma},$$

其中 $\psi(x)=\operatorname{sign}(x)\max\{|x|-1,0\}$, $\operatorname{sign}(x)$ 表示 x 的符号，即

$$\operatorname{sign}(x)=\begin{cases}1, & x>0,\\ 0, & x=0,\\ -1, & x<0.\end{cases}$$

注意，为了记号简洁，我们仍然使用 L_σ 来表示增广拉格朗日函数，但变量个数有所变化.

消去 s 的增广拉格朗日函数法为：

$$\begin{cases}y^{k+1}=\arg\min\limits_y\left\{b^{\mathrm{T}}y+\dfrac{\sigma}{2}\left\|\psi\left(A^{\mathrm{T}}y+\dfrac{\lambda^k}{\sigma_k}\right)\right\|_2^2\right\},\\ \lambda^{k+1}=\sigma_k\psi\left(A^{\mathrm{T}}y^{k+1}+\dfrac{\lambda^k}{\sigma_k}\right),\\ \sigma_{k+1}=\min\{\rho\sigma_k,\bar{\sigma}\}.\end{cases}\tag{6.2.26}$$

在迭代格式(6.2.26)的第一步中，我们不能得到关于 y^{k+1} 的显式表达式. 但是由于 $L_{\sigma_k}(y,\lambda^k)$ 关于 y 是连续可微的，且其梯度为

$$\nabla_y L_{\sigma_k}(y,\lambda^k)=b+\sigma_k A\psi\left(A^{\mathrm{T}}y+\frac{\lambda^k}{\sigma_k}\right).$$

可以利用梯度法对其进行求解. 除此之外，还可以采用半光滑牛顿法，相关内容可以参考 [78,141].

记 $\phi_k(y)=L_{\sigma_k}(y,\lambda^k)$. 为了保证收敛性，根据一般凸优化问题的增广拉格朗日函数法的收敛条件 (6.2.15)，我们要求 y^{k+1} 满足

$$\phi_k(y^{k+1})-\inf\phi_k\leqslant\frac{\varepsilon_k^2}{2\sigma_k},\quad \varepsilon_k\geqslant 0,\sum_{k=1}^{\infty}\varepsilon_k+\infty,\tag{6.2.27}$$

其中 ε_k 是人为设定的参数.

根据定理 6.7，有如下收敛性定理.

定理 6.8 假设 $\{y^k\},\{\lambda^k\}$ 是由迭代格式 (6.2.26) 产生的序列，并且 y^{k+1} 的求解精度满足 (6.2.27) 式，而矩阵 A 是行满秩的. 那么，序列 $\{y^k\}$ 是有界的，且其任一聚点均为问题 (6.2.21) 的最优解. 同时，序列 $\{\lambda^k\}$ 有界且收敛，其极限为原始问题 (6.2.19) 的某个最优解.

注 6.3 定理 6.8 假设 A 是行满秩的，因此，我们知道可行域

$$\mathcal{X} = \{y | \|A^{\mathrm{T}} y\|_\infty \leqslant 1\}$$

是有界的. 由于 $0 \in \mathcal{X}$，故 $\{x \in \mathcal{X} \mid f(x) \leqslant 0\}$ 是非空有界的. 根据约束的线性性，易知问题 (6.2.20) 的 Slater 约束品性成立.

这里注意，ϕ_k 只是凸的，并不是强凸的.我们可以通过添加 $\dfrac{1}{2\sigma_k}\|y-y^k\|_2^2$，并求解

$$y^{k+1} \approx \arg\min_y \left\{ \phi_k(y) + \frac{1}{2\sigma_k}\|y-y^k\|_2^2 \right\}$$

使得 y^{k+1} 满足不精确条件 (6.2.27). 此时，函数 $\phi_k(y) + \dfrac{1}{2\sigma_k}\|y-y^k\|_2^2$ 是 $\dfrac{1}{\sigma_k}$ 强凸的. 修改后的迭代点列的收敛性基本与原始问题增广拉格朗日函数法的一致，证明细节可以参考 [79,141].

6.3 总结

本章介绍了一般约束优化问题的罚函数法和增广拉格朗日函数法. 相较于罚函数法，增广拉格朗日函数法具有更好的理论性质（尤其是对凸优化问题）. 关于凸优化问题的增广拉格朗日函数法，读者可以进一步参考 [111]. 我们知道增广拉格朗日函数法的困难之一是决策变量的更新. 除了前面介绍的利用半光滑性质，还可以利用交替方向乘子法，其给出了一种有效的更新方式，我们会在第七章中详细介绍该方法.

本章的罚函数法、一般优化问题的增广拉格朗日函数法相关内容的编写参考了 [98]，凸优化问题、基追踪问题的增广拉格朗日函数法的编写参考了 [111,133-134,141].

习题 7

6.1 构造一个等式约束优化问题，使得它存在一个局部极小值，但对于任意的 $\sigma > 0$，它的二次罚函数是无界的.

6.2 考虑等式约束优化问题

$$\begin{aligned} \min \quad & -x_1x_2x_3, \\ \text{s.t.} \quad & x_1+2x_2+3x_3=60. \end{aligned}$$

使用二次罚函数求解该问题，当固定罚因子 σ_k 时，写出二次罚函数的最优解 x^{k+1}. 当 $\sigma_k\to+\infty$ 时，写出该优化问题的解并求出约束的拉格朗日乘子. 此外，当罚因子 σ 满足什么条件时，二次罚函数的海瑟矩阵 $\nabla^2_{xx}P_E(x,\sigma)$ 是正定的?

6.3 考虑等式约束优化问题

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

定义一般形式的罚函数

$$P_E(x,\sigma)=f(x)+\sigma\sum_{i\in\mathcal{E}}\varphi(c_i(x)),$$

其中 $\varphi(t)$ 是充分光滑的函数，且 $t=0$ 是其 s 阶零点（$s\geqslant 2$），即

$$\varphi(0)=\varphi'(0)=\cdots=\varphi^{(s-1)}(0)=0, \quad \varphi^{(s)}(0)\neq 0.$$

设 x^k,σ_k 的选取方式和算法 6.1 的相同，且 $\{x^k\}$ 存在极限 x^*，在点 x^* 处 LICQ （见定义 4.7）成立.

(a) 证明: $\sigma_k(c_i(x^k))^{s-1},\forall\, i\in\mathcal{E}$ 极限存在，其极限 λ_i^* 为约束 $c_i(x^*)=0$ 对应的拉格朗日乘子;

(b) 求 $P_E(x,\sigma)$ 关于 x 的海瑟矩阵 $\nabla^2_{xx}P_E(x,\sigma)$;

(c) 设在 (a) 中 $\lambda_i^*\neq 0,\forall\, i\in\mathcal{E}$，证明: 当 $\sigma_k\to+\infty$ 时，$\nabla^2_{xx}P_E(x^k,\sigma_k)$ 有 m 个特征值的模长与 $\sigma_k^{1/(s-1)}$ 同阶，其中 $m=|\mathcal{E}|$.

6.4 考虑不等式约束优化问题 (6.1.12)，其中 f 在可行域 $\mathcal{X}$ 上有下界，现使用对数罚函数法进行求解（算法 6.3）. 假设在算法 6.3 的每一步子问题能求出罚函数的全局极小值点 x^{k+1}，证明: 算法 6.3 在有限次迭代后终止，或者

$$\lim_{k\to\infty}\sigma_k\sum_{i\in\mathcal{I}}\ln(-c_i(x^{k+1}))=0,$$

并且

$$\lim_{k\to\infty}f(x^k)=\inf_{x\in\mathbf{int}\mathcal{X}}f(x).$$

6.5 考虑一般约束优化问题 (6.1.15)，现在针对等式约束使用二次罚函数，对不等式约束使用对数罚函数：

$$P(x,\sigma)=f(x)+\frac{\sigma}{2}\sum_{i\in\mathcal{E}}c_i^2(x)-\frac{1}{\sigma}\sum_{i\in\mathcal{I}}\ln(-c_i(x)),$$

其中 $\mathbf{dom}\,P=\{x\mid c_i(x)<0,\,i\in\mathcal{I}\}$. 令罚因子 $\sigma_k\to+\infty$，定义

$$x^{k+1}=\arg\min_x P(x,\sigma_k).$$

假定涉及的所有函数都是连续的，$\{x\mid c_i(x)\leqslant 0,\,i\in\mathcal{I}\}$ 是有界闭集，x^* 为问题 (6.1.15) 的解. 试证明如下结论：

(a) $\lim\limits_{k\to\infty}P(x^{k+1},\sigma_k)=f(x^*)$;

(b) $\lim\limits_{k\to\infty}\sigma_k\sum\limits_{i\in\mathcal{E}}c_i^2(x^{k+1})=0$;

(c) $\lim\limits_{k\to\infty}\dfrac{1}{\sigma_k}\sum\limits_{i\in\mathcal{I}}\ln(-c_i(x^{k+1}))=0$.

6.6 (Morrison 方法) 考虑等式约束优化问题 (6.1.1)，设其最优解为 x^*. 令 M 是最优函数值 $f(x^*)$ 的一个下界估计（即 $M\leqslant f(x^*)$），构造辅助函数

$$v(M,x)=[f(x)-M]^2+\sum_{i\in\mathcal{E}}c_i^2(x),$$

Morrison 方法的迭代步骤如下：

$$x^k=\arg\min_x\ v(M_k,x),$$

$$M_{k+1}=M_k+\sqrt{v(M_k,x^k)}.$$

试回答以下问题：

(a) 证明：$f(x^k)\leqslant f(x^*)$;

(b) 若 $M_k\leqslant f(x^*)$，证明：$M_{k+1}\leqslant f(x^*)$;

(c) 证明：$\lim\limits_{k\to\infty}M_k=f(x^*)$;

(d) 求 $v(M,x)$ 关于 x 的海瑟矩阵，并说明 Morrison 方法和算法 6.1 的联系.

6.7 考虑不等式约束优化问题

$$\min\quad f(x),\quad \text{s.t.}\quad c_i(x)\leqslant 0,\,i\in\mathcal{I}.$$

(a) 定义函数 $F(x)=\sup\limits_{\lambda_i\geqslant 0}\left\{f(x)+\sum\limits_{i\in\mathcal{I}}\lambda_i c_i(x)\right\}$，证明：原问题等价于无约束优化问题 $\min\limits_x F(x)$；

(b) 定义函数

$$\hat{F}(x,\lambda^k,\sigma_k)=\sup_{\lambda_i\geqslant 0}\left\{f(x)+\sum_{i\in\mathcal{I}}\lambda_i c_i(x)-\frac{\sigma_k}{2}\sum_{i\in\mathcal{I}}(\lambda_i-\lambda_i^k)^2\right\},$$

求 $\hat{F}(x,\lambda^k,\sigma_k)$ 的显式表达式；

(c) 考虑如下优化算法：

$$\begin{aligned}x^k&=\arg\min_x \hat{F}(x,\lambda^k,\sigma_k),\\ \lambda^{k+1}&=\arg\max_{\lambda\geqslant 0}\left\{\sum_{i\in\mathcal{I}}\lambda_i c_i(x^k)-\frac{\sigma_k}{2}\sum_{i\in\mathcal{I}}(\lambda_i-\lambda_i^k)^2\right\},\\ \sigma_{k+1}&=\min\{\rho\sigma_k,\bar{\sigma}\},\end{aligned}$$

试说明其与算法 6.4 的区别和联系.

6.8 对于 LASSO 问题

$$\min_{x\in\mathbb{R}^n}\quad \frac{1}{2}\|Ax-b\|_2^2+\mu\|x\|_1,$$

写出该问题及其对偶问题的增广拉格朗日函数法.

6.9 考虑线性规划问题

$$\min_{x\in\mathbb{R}^n}\quad c^{\mathrm{T}}x,\quad \text{s.t.}\quad Ax=b,\ x\geqslant 0.$$

写出该问题以及其对偶问题的增广拉格朗日函数法.

第七章　复合优化算法

本章主要考虑如下复合优化问题：

$$\min_{x\in\mathbb{R}^n} \quad \psi(x) \xlongequal{\text{def}} f(x) + h(x), \tag{7.0.1}$$

其中 $f(x)$ 为可微函数（可能非凸），$h(x)$ 可能为不可微函数. 问题 (7.0.1) 出现在很多应用领域中，例如压缩感知、图像处理、机器学习等，如何高效求解该问题是近年来的热门课题. 第五章曾利用光滑化的思想处理不可微项 $h(x)$，但这种做法没有充分利用 $h(x)$ 的性质，在实际应用中有一定的局限性. 而本章将介绍若干适用于求解问题 (7.0.1) 的方法并给出一些理论性质. 我们首先引入针对问题 (7.0.1) 直接进行求解的近似点梯度法和 Nesterov 加速算法，之后介绍求解特殊结构复合优化问题的近似点算法、分块坐标下降法、对偶算法以及交替方向乘子法，最后介绍处理 $\nabla f(x)$ 难以精确计算情形的随机优化算法.

需要注意的是，许多实际问题并不直接具有本章介绍的算法所能处理的形式，我们需要利用拆分、引入辅助变量等技巧将其进行等价变形，最终化为合适的优化问题. 具体可参考第 7.4 节和第 7.6 节的内容. 此外，本章涉及的定理证明需要较多第二章中的内容，为了方便，我们默认所有次梯度计算规则的前提成立（加法、线性变量替换等，见第 2.7.3 节）. 这些前提在绝大多数应用中都会满足.

7.1　近似点梯度法

在机器学习、图像处理领域中，许多模型包含两部分：一部分是误差项，一般为光滑函数；另外一部分是正则项，可能为非光滑函数，用来保证求解问题的特殊结构. 例如最常见的 LASSO 问题就是用 ℓ_1 范数构造正则项保证求解的参数是稀疏的，从而起到筛选变量的作用. 由于有非光滑部分的存

在，此类问题属于非光滑的优化问题，我们可以考虑使用次梯度算法进行求解. 然而次梯度算法并不能充分利用光滑部分的信息，也很难在迭代中保证非光滑项对应的解的结构信息，这使得次梯度算法在求解这类问题时往往收敛较慢. 本节将介绍求解这类问题非常有效的一种算法——近似点梯度算法. 它能克服次梯度算法的缺点，充分利用光滑部分的信息，并在迭代过程中显式地保证解的结构，从而能够达到和求解光滑问题的梯度算法相近的收敛速度. 在后面的内容中，我们首先引入邻近算子，它是近似点梯度算法中处理非光滑部分的关键；接着介绍近似点梯度算法的迭代格式，并给出一些实际的例子；最后给出这个算法的一些收敛性证明，并将看到它确实有和光滑梯度算法相似的收敛速度. 为了讨论简便，我们主要介绍凸函数的情形.

7.1.1 邻近算子

邻近算子是处理非光滑问题的一个非常有效的工具，也与许多算法的设计密切相关，比如我们即将介绍的近似点梯度法和近似点算法等. 当然该算子并不局限于非光滑函数，也可以用来处理光滑函数. 本小节将介绍邻近算子的相关内容，为引入近似点梯度算法做准备.

首先给出邻近算子的定义.

定义 7.1 (邻近算子) 对于一个凸函数 h，定义它的邻近算子为

$$\mathrm{prox}_h(x) = \underset{u\in\mathbf{dom}\,h}{\arg\min}\left\{h(u) + \frac{1}{2}\|u-x\|^2\right\}. \tag{7.1.1}$$

可以看到，邻近算子的目的是求解一个距 x 不算太远的点，并使函数值 $h(x)$ 也相对较小. 一个很自然的问题是，上面给出的邻近算子的定义是不是有意义的，即定义中的优化问题的解是不是存在唯一的. 若答案是肯定的，我们就可使用邻近算子去构建迭代格式. 下面的定理将给出定义中优化问题解的存在唯一性.

定理 7.1 (邻近算子是良定义的) 如果 h 是适当的闭凸函数，则对任意的 $x\in\mathbb{R}^n$，$\mathrm{prox}_h(x)$ 的值存在且唯一.

证明. 为了简化证明，我们假设 h 至少在定义域内的一点处存在次梯度，保证次梯度存在的一个充分条件是 $\mathbf{dom}\,h$ 内点集非空. 对于比较复杂的情况读者可参考 [6]命题 12.15. 定义辅助函数

$$m(u) = h(u) + \frac{1}{2}\|u-x\|^2,$$

下面利用 Weierstrass 定理（定理 4.1）来说明 $m(u)$ 最小值点的存在性. 因为 $h(u)$ 是凸函数，且至少在一点处存在次梯度，所以 $h(u)$ 有全局下界：

$$h(u) \geqslant h(v) + \theta^{\mathrm{T}}(u - v),$$

这里 $v \in \mathbf{dom}\, h$，$\theta \in \partial h(v)$. 进而得到

$$\begin{aligned} m(u) &= h(u) + \frac{1}{2}\|u - x\|^2 \\ &\geqslant h(v) + \theta^{\mathrm{T}}(u - v) + \frac{1}{2}\|u - x\|^2, \end{aligned}$$

这表明 $m(u)$ 具有二次下界. 容易验证 $m(u)$ 为适当闭函数且具有强制性（当 $\|u\| \to +\infty$ 时，$m(u) \to +\infty$），根据定理 4.1 可知 $m(u)$ 存在最小值.

接下来证明唯一性. 注意到 $m(u)$ 是强凸函数，根据命题 2.3 的结果可直接得出 $m(u)$ 的最小值唯一. 综上 $\mathrm{prox}_h(x)$ 是良定义的. □

另外，根据最优性条件可以得到如下等价结论：

定理 7.2 (邻近算子与次梯度的关系) *如果 h 是适当的闭凸函数，则*

$$u = \mathrm{prox}_h(x) \iff x - u \in \partial h(u).$$

证明. 若 $u = \mathrm{prox}_h(x)$，则由最优性条件得 $0 \in \partial h(u) + (u - x)$，因此有 $x - u \in \partial h(u)$.

反之，若 $x - u \in \partial h(u)$ 则由次梯度的定义可得到

$$h(v) \geqslant h(u) + (x - u)^{\mathrm{T}}(v - u), \quad \forall\, v \in \mathbf{dom}\, h.$$

两边同时加 $\frac{1}{2}\|v - x\|^2$，即有

$$\begin{aligned} h(v) + \frac{1}{2}\|v - x\|^2 &\geqslant h(u) + (x - u)^{\mathrm{T}}(v - u) + \frac{1}{2}\|v - x\|^2 \\ &\geqslant h(u) + \frac{1}{2}\|u - x\|^2, \quad \forall\, v \in \mathbf{dom}\, h. \end{aligned}$$

因此我们得到 $u = \mathrm{prox}_h(x)$. □

用 th 代替 h，上面的等价结论形式上可以写成

$$u = \mathrm{prox}_{th}(x) \iff u \in x - t\partial h(u).$$

邻近算子的计算可以看成是次梯度算法的隐式格式（后向迭代），这实际是近似点算法的迭代格式（见第7.3节）. 对于非光滑情形，由于次梯度不唯一，显式格式的迭代并不唯一，而隐式格式却能得到唯一解. 此外在步长的选择上面，隐式格式也要优于显式格式.

下面给出一些常见的例子. 计算邻近算子的过程实际上是在求解一个优化问题，我们给出 ℓ_1 范数和 ℓ_2 范数对应的计算过程，其他例子读者可以自行验证.

例 7.1 (邻近算子的例子) 在下面所有例子中，常数 $t>0$ 为正实数.

(1) ℓ_1 范数：

$$h(x)=\|x\|_1,\quad \mathrm{prox}_{th}(x)=\mathrm{sign}(x)\max\{|x|-t,0\}.$$

证明. 邻近算子 $u=\mathrm{prox}_{th}(x)$ 的最优性条件为

$$x-u\in t\partial\|u\|_1=\begin{cases}\{t\}, & u>0,\\ [-t,t], & u=0,\\ \{-t\}, & u<0,\end{cases}$$

因此，当 $x>t$ 时，$u=x-t$；当 $x<-t$ 时，$u=x+t$；当 $x\in[-t,t]$ 时，$u=0$，即有 $u=\mathrm{sign}(x)\max\{|x|-t,0\}$. □

(2) ℓ_2 范数：

$$h(x)=\|x\|_2,\quad \mathrm{prox}_{th}(x)=\begin{cases}\left(1-\dfrac{t}{\|x\|_2}\right)x, & \|x\|_2\geqslant t,\\ 0, & \text{其他}.\end{cases}$$

证明. 邻近算子 $u=\mathrm{prox}_{th}(x)$ 的最优性条件为

$$x-u\in t\partial\|u\|_2=\begin{cases}\left\{\dfrac{tu}{\|u\|_2}\right\}, & u\neq 0,\\ \{w:\|w\|_2\leqslant t\}, & u=0,\end{cases}$$

因此，当 $\|x\|_2>t$ 时，$u=x-\dfrac{tx}{\|x\|_2}$；当 $\|x\|_2\leqslant t$ 时，$u=0$. □

除了直接利用定义计算，很多时候可以利用已知邻近算子的结果，计算其他邻近算子. 下面我们给出一些常用的运算规则. 运用这些简单的规则，可以求解更复杂的邻近算子.

例 7.2 (邻近算子的运算规则) 由邻近算子的定义和基本的计算推导，我们可以得出邻近算子满足如下运算规则：

(1) 变量的常数倍放缩以及平移 ($\lambda \neq 0$)：

$$h(x) = g(\lambda x + a), \quad \mathrm{prox}_h(x) = \frac{1}{\lambda}(\mathrm{prox}_{\lambda^2 g}(\lambda x + a) - a);$$

(2) 函数（及变量）的常数倍放缩 ($\lambda > 0$)：

$$h(x) = \lambda g\left(\frac{x}{\lambda}\right), \quad \mathrm{prox}_h(x) = \lambda \mathrm{prox}_{\lambda^{-1} g}\left(\frac{x}{\lambda}\right);$$

(3) 加上线性函数：

$$h(x) = g(x) + a^{\mathrm{T}} x, \quad \mathrm{prox}_h(x) = \mathrm{prox}_g(x - a);$$

(4) 加上二次项 ($u > 0$)

$$h(x) = g(x) + \frac{u}{2}\|x - a\|_2^2, \quad \mathrm{prox}_h(x) = \mathrm{prox}_{\theta g}(\theta x + (1 - \theta)a);$$

其中 $\theta = \dfrac{1}{1+u}$;

(5) 向量函数：

$$h\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \varphi_1(x) + \varphi_2(y), \quad \mathrm{prox}_h\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} \mathrm{prox}_{\varphi_1}(x) \\ \mathrm{prox}_{\varphi_2}(y) \end{bmatrix}.$$

另外一种比较常用的邻近算子是关于示性函数的邻近算子. 集合 C 的示性函数定义为

$$I_C(x) = \begin{cases} 0, & x \in C, \\ +\infty, & \text{其他}, \end{cases}$$

它可以用来把约束变成目标函数的一部分.

例 7.3 (闭凸集上的投影) 设 C 为 $\mathbb{R}^n$ 上的闭凸集，则示性函数 I_C 的邻近算子为点 x 到集合 C 的投影，即

$$\begin{aligned} \mathrm{prox}_{I_C}(x) &= \arg\min_u \left\{ I_C(u) + \frac{1}{2}\|u - x\|^2 \right\} \\ &= \arg\min_{u \in C} \|u - x\|^2 = \mathcal{P}_C(x). \end{aligned}$$

此外，应用定理 7.2 可进一步得到

$$\begin{aligned} u = \mathcal{P}_C(x) &\Leftrightarrow x - u \in \partial I_C(u) \\ &\Leftrightarrow (x-u)^{\mathrm{T}}(z-u) \leqslant I_C(z) - I_C(u) = 0, \quad \forall\, z \in C. \end{aligned}$$

此结论有较强的几何意义：若点 x 位于 C 外部，则从投影点 u 指向 x 的向量与任意起点为 u 且指向 C 内部的向量的夹角为直角或钝角.

7.1.2 近似点梯度法

下面将引入本节的重点——近似点梯度算法. 我们将考虑如下复合优化问题：

$$\min \quad \psi(x) = f(x) + h(x), \tag{7.1.2}$$

其中函数 f 为可微函数，其定义域 $\mathbf{dom}\, f = \mathbb{R}^n$，函数 h 为凸函数，可以是非光滑的，并且一般计算此项的邻近算子并不复杂. 比如 LASSO 问题，两项分别为 $f(x) = \dfrac{1}{2}\|Ax-b\|^2$，$h(x) = \mu\|x\|_1$. 一般的带凸集约束的优化问题也可以用 (7.1.2) 式表示，即对问题

$$\min_{x \in C} \quad \phi(x),$$

复合优化问题中的两项可以写作 $f(x) = \phi(x)$，$h(x) = I_C(x)$，其中 $I_C(x)$ 为示性函数.

近似点梯度法的思想非常简单：注意到 $\psi(x)$ 有两部分，对于光滑部分 f 做梯度下降，对于非光滑部分 h 使用邻近算子，则近似点梯度法的迭代公式为

$$x^{k+1} = \mathrm{prox}_{t_k h}(x^k - t_k \nabla f(x^k)), \tag{7.1.3}$$

其中 $t_k > 0$ 为每次迭代的步长，它可以是一个常数或者由线搜索得出. 近似点梯度法跟众多算法都有很强的联系，在一些特定条件下，近似点梯度法还可以转化为其他算法：当 $h(x) = 0$ 时，迭代公式变为梯度下降法

$$x^{k+1} = x^k - t_k \nabla f(x^k);$$

当 $h(x) = I_C(x)$ 时，迭代公式变为投影梯度法

$$x^{k+1} = \mathcal{P}_C(x^k - t_k \nabla f(x^k)).$$

近似点梯度法可以总结为算法 7.1.

算法 7.1 近似点梯度法

1. **输入**：函数 $f(x), h(x)$，初始点 x^0. 初始化 $k=0$.
2. **while** 未达到收敛准则 **do**
3. $\quad x^{k+1} = \operatorname{prox}_{t_k h}(x^k - t_k \nabla f(x^k))$.
4. $\quad k \leftarrow k+1$.
5. **end while**

如何理解近似点梯度法？根据邻近算子的定义，把迭代公式展开：

$$\begin{aligned} x^{k+1} &= \arg\min_u \left\{ h(u) + \frac{1}{2t_k}\|u - x^k + t_k \nabla f(x^k)\|^2 \right\} \\ &= \arg\min_u \left\{ h(u) + f(x^k) + \nabla f(x^k)^{\mathrm{T}}(u - x^k) + \frac{1}{2t_k}\|u - x^k\|^2 \right\}, \end{aligned}$$

可以发现，近似点梯度法实质上就是将问题的光滑部分线性展开再加上二次项并保留非光滑部分，然后求极小来作为每一步的估计. 此外，根据定理 7.2，近似点梯度算法可以形式上写成

$$x^{k+1} = x^k - t_k \nabla f(x^k) - t_k g^k, \quad g^k \in \partial h(x^{k+1}).$$

其本质上是对光滑部分做显式的梯度下降，关于非光滑部分做隐式的梯度下降.

算法 7.1 中步长 t_k 的选取较为关键. 当 f 为梯度 L-利普希茨连续函数时，可取固定步长 $t_k = t \leqslant \dfrac{1}{L}$. 当 L 未知时可使用线搜索准则

$$f(x^{k+1}) \leqslant f(x^k) + \nabla f(x^k)^{\mathrm{T}}(x^{k+1} - x^k) + \frac{1}{2t_k}\|x^{k+1} - x^k\|^2. \tag{7.1.4}$$

我们将在第 7.1.4 小节中解释这样选取的原因. 此外，还可利用 BB 步长作为 t_k 的初始估计并用非单调线搜索准则进行校正. 由于 $\psi(x)$ 是不可微的函数，利用格式 (5.2.7) 或格式 (5.2.8) 进行计算时，应使用 $\nabla f(x^k)$ 和 $\nabla f(x^{k-1})$（即光滑部分的梯度）计算与其对应的 y^{k-1}. 类似地，仿照准则 (5.1.6) 可构造如下适用于近似点梯度法的非单调线搜索准则：

$$\psi(x^{k+1}) \leqslant C^k - \frac{c_1}{2t_k}\|x^{k+1} - x^k\|^2, \tag{7.1.5}$$

其中 $c_1 \in (0,1)$ 为正常数，C^k 的定义同 (5.1.6) 式. 注意，定义 C^k 时需要使用整体函数值 $\psi(x^k)$.

7.1.3 应用举例

1. LASSO 问题求解

这里介绍如何使用近似点梯度法来求解 LASSO 问题

$$\min_x \quad \mu\|x\|_1 + \frac{1}{2}\|Ax-b\|^2.$$

令 $f(x)=\frac{1}{2}\|Ax-b\|^2$，$h(x)=\mu\|x\|_1$，则

$$\nabla f(x) = A^{\mathrm{T}}(Ax-b),$$
$$\mathrm{prox}_{t_k h}(x) = \mathrm{sign}(x)\max\{|x|-t_k\mu, 0\}.$$

求解 LASSO 问题的近似点梯度算法可以由下面的迭代格式给出：

$$y^k = x^k - t_k A^{\mathrm{T}}(Ax^k-b),$$
$$x^{k+1} = \mathrm{sign}(y^k)\max\{|y^k|-t_k\mu, 0\},$$

即第一步做梯度下降，第二步做收缩．特别地，第二步收缩算子保证了迭代过程中解的稀疏结构．这也解释了为什么近似点梯度算法效果好的原因．

我们用同第 5.2 节中一样的 A 和 b，并取 $\mu=10^{-3}$．采用连续化的近似点梯度法来求解，分别取固定步长 $t=\frac{1}{L}$，这里 $L=\lambda_{max}(A^{\mathrm{T}}A)$，和结合线搜索的 BB 步长．停机准则和参数 μ 的连续化设置和第 5.2 节中的光滑化梯度法一致，结果如图7.1.

可以看到结合线搜索的 BB 步长能够显著提高算法的收敛速度，且比第5.2节中的光滑化梯度法收敛得更快.

2. 低秩矩阵恢复

考虑低秩矩阵恢复模型：

$$\min_{X\in\mathbb{R}^{m\times n}} \quad \mu\|X\|_* + \frac{1}{2}\sum_{(i,j)\in\Omega}(X_{ij}-M_{ij})^2,$$

其中 M 是想要恢复的低秩矩阵，但是只知道其在下标集 Ω 上的值．令

$$f(X)=\frac{1}{2}\sum_{(i,j)\in\Omega}(X_{ij}-M_{ij})^2, \quad h(X)=\mu\|X\|_*,$$

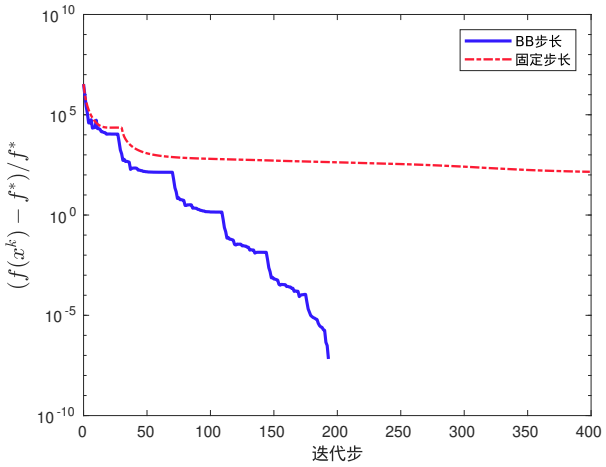


图 7.1 近似点梯度法求解 LASSO 问题

定义矩阵 $P \in \mathbb{R}^{m\times n}$:

$$P_{ij} = \begin{cases} 1, & (i,j) \in \Omega, \\ 0, & \text{其他}, \end{cases}$$

则

$$\begin{aligned} f(X) &= \frac{1}{2}\|P \odot (X - M)\|_F^2, \\ \nabla f(X) &= P \odot (X - M), \\ \text{prox}_{t_k h}(X) &= U\text{Diag}\left(\max\{|d| - t_k\mu, 0\}\right) V^{\mathrm{T}}, \end{aligned}$$

其中 $X = U\text{Diag}(d)V^{\mathrm{T}}$ 为矩阵 X 的约化的奇异值分解. 近似点梯度法的迭代格式为

$$\begin{aligned} Y^k &= X^k - t_k P \odot (X^k - M), \\ X^{k+1} &= \text{prox}_{t_k h}(Y^k). \end{aligned}$$

7.1.4 收敛性分析

本小节介绍近似点梯度算法的收敛性. 在提出近似点梯度算法时, 我们仅仅要求 $f(x)$ 为可微函数, 但在收敛性分析时则要求 $f(x)$ 也为凸函数.

假设 7.1

(1) f 在其定义域 $\mathbf{dom} f = \mathbb{R}^n$ 内为凸的；∇f 在常数 L 意义下利普希茨连续，即

$$\|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\|, \quad \forall x, y;$$

(2) h 是适当的闭凸函数（因此 prox_{th} 的定义是合理的）；

(3) 函数 $\psi(x) = f(x) + h(x)$ 的最小值 ψ^* 是有限的，并且在点 x^* 处可以取到（并不要求唯一）.

以上的条件可以保证近似点梯度法的收敛结果: 在定步长 $t_k \in \left(0, \frac{1}{L}\right]$ 的情况下，迭代点 x^k 处的函数值 $\psi(x^k)$ 以 $\mathcal{O}\left(\frac{1}{k}\right)$ 的速率收敛到 ψ^*.

在正式给出收敛定理之前，我们先引入一个新函数.

定义 7.2 (梯度映射) 设 $f(x)$ 和 $h(x)$ 满足假设 7.1，$t > 0$ 为正常数，定义梯度映射为

$$G_t(x) = \frac{1}{t}(x - \mathrm{prox}_{th}(x - t\nabla f(x))). \tag{7.1.6}$$

通过计算可以发现 $G_t(x)$ 为近似点梯度法每次迭代中的负的“搜索方向”，即

$$x^{k+1} = \mathrm{prox}_{th}(x^k - t\nabla f(x^k)) = x^k - tG_t(x^k).$$

这里需要注意的是，$G_t(x)$ 并不是 $\psi = f + h$ 的梯度或者次梯度，而由之前邻近算子与次梯度的关系可以得出

$$G_t(x) - \nabla f(x) \in \partial h(x - tG_t(x)). \tag{7.1.7}$$

此外，$G_t(x)$ 作为“搜索方向”，与算法的收敛性有很强的关系：$G_t(x) = 0$ 当且仅当 x 为 $\psi(x) = f(x) + h(x)$ 的最小值点.

有了上面的铺垫，我们介绍近似点梯度法的收敛性.

定理 7.3 在假设 7.1下，取定步长为 $t_k = t \in \left(0, \frac{1}{L}\right]$，设 $\{x^k\}$ 是由迭代格式(7.1.3) 产生的序列，则

$$\psi(x^k) - \psi^* \leqslant \frac{1}{2kt}\|x^0 - x^*\|^2. \tag{7.1.8}$$

证明. 利用假设 7.1 中的利普希茨连续的性质，根据二次上界(2.2.3)可以得到

$$f(y) \leqslant f(x) + \nabla f(x)^{\mathrm{T}}(y - x) + \frac{L}{2}\|y - x\|^2, \quad \forall x, y \in \mathbb{R}^n.$$

令 $y = x - tG_t(x)$ ，有

$$f(x - tG_t(x)) \leqslant f(x) - t\nabla f(x)^{\mathrm{T}} G_t(x) + \frac{t^2 L}{2}\|G_t(x)\|^2.$$

若 $0 < t \leqslant \dfrac{1}{L}$, 则

$$f(x - tG_t(x)) \leqslant f(x) - t\nabla f(x)^{\mathrm{T}} G_t(x) + \frac{t}{2}\|G_t(x)\|^2. \tag{7.1.9}$$

此外，由 $f(x), h(x)$ 为凸函数，对任意 $z \in \mathbf{dom}\,\psi$ 我们有

$$h(z) \geqslant h(x - tG_t(x)) + (G_t(x) - \nabla f(x))^{\mathrm{T}}(z - x + tG_t(x)),$$
$$f(z) \geqslant f(x) + \nabla f(x)^{\mathrm{T}}(z - x),$$

其中关于 $h(z)$ 的不等式利用了关系式(7.1.7). 整理得

$$h(x - tG_t(x)) \leqslant h(z) - (G_t(x) - \nabla f(x))^{\mathrm{T}}(z - x + tG_t(x)), \tag{7.1.10}$$
$$f(x) \leqslant f(z) - \nabla f(x)^{\mathrm{T}}(z - x). \tag{7.1.11}$$

将(7.1.9)—(7.1.11)式相加可得对任意 $z \in \mathbf{dom}\,\psi$ 有

$$\psi(x - tG_t(x)) \leqslant \psi(z) + G_t(x)^{\mathrm{T}}(x - z) - \frac{t}{2}\|G_t(x)\|^2. \tag{7.1.12}$$

因此，对于每一步的迭代，

$$\tilde{x} = x - tG_t(x),$$

在全局不等式 (7.1.12) 中，取 $z = x^*$ 有

$$\begin{aligned}\psi(\tilde{x}) - \psi^* &\leqslant G_t(x)^{\mathrm{T}}(x - x^*) - \frac{t}{2}\|G_t(x)\|^2 \\ &= \frac{1}{2t}\left(\|x - x^*\|^2 - \|x - x^* - tG_t(x)\|^2\right) \\ &= \frac{1}{2t}\left(\|x - x^*\|^2 - \|\tilde{x} - x^*\|^2\right).\end{aligned} \tag{7.1.13}$$

分别取 $x = x^{i-1}, \tilde{x} = x^i, t = t_i = \dfrac{1}{L}, i = 1, 2, \cdots, k$，代入不等式 (7.1.13) 并累加，

$$\begin{aligned}\sum_{i=1}^{k}(\psi(x^i) - \psi^*) &\leqslant \frac{1}{2t}\sum_{i=1}^{k}\left(\|x^{i-1} - x^*\|^2 - \|x^i - x^*\|^2\right) \\ &= \frac{1}{2t}\left(\|x^0 - x^*\|^2 - \|x^k - x^*\|^2\right) \\ &\leqslant \frac{1}{2t}\|x^0 - x^*\|^2.\end{aligned}$$

注意到在不等式 (7.1.12) 中，取 $z = x$ 即可得到算法为下降法：

$$\psi(\tilde{x}) \leqslant \psi(x) - \frac{t}{2}\|G_t(x)\|^2,$$

即 $\psi(x^i)$ 为非增的，因此

$$\psi(x^k) - \psi^* \leqslant \frac{1}{k}\sum_{i=1}^{k}(\psi(x^i) - \psi^*) \leqslant \frac{1}{2kt}\|x^0 - x^*\|^2. \qquad \square$$

在定理 7.3 中，收敛性要求步长小于或等于 ∇f 对应的利普希茨常数 L 的倒数. 但是在实际应用中，我们经常很难知道 L，因此可以考虑线搜索的技巧. 注意到之所以需要 $t \leqslant \frac{1}{L}$ 的条件是因为不等式(7.1.9)，所以线搜索策略可以是从某个 $t = \hat{t} > 0$ 开始进行回溯（$t \leftarrow \beta t$），直到满足不等式

$$f(x - tG_t(x)) \leqslant f(x) - t\nabla f(x)^{\mathrm{T}}G_t(x) + \frac{t}{2}\|G_t(x)\|^2. \tag{7.1.14}$$

注意，(7.1.14) 式与前面给出的线搜索准则 (7.1.4) 是等价的，这也说明了准则 (7.1.4) 的合理性.

类似于定理 7.3，我们有如下收敛性定理：

定理 7.4 *在假设 7.1下，从某个 $t = \hat{t} > 0$ 开始进行回溯（$t \leftarrow \beta t$）直到满足不等式(7.1.14)，设 $\{x^k\}$ 是由迭代格式(7.1.3) 产生的序列，则*

$$\psi(x^k) - \psi^* \leqslant \frac{1}{2k\min\{\hat{t}, \beta/L\}}\|x^0 - x^*\|^2.$$

证明. 由定理 7.3 的证明过程，当 $0 < t \leqslant \frac{1}{L}$ 时，不等式(7.1.14)成立，因此由线搜索所得的步长 t 应该满足 $t \geqslant t_{\min} = \min\{\hat{t}, \frac{\beta}{L}\}$. 利用和定理 7.3 同样的证明方法，我们有 $\psi(x^i) < \psi(x^{i-1})$，且

$$\psi(x^i) - \psi^* \leqslant \frac{1}{2t_{\min}}\left(\|x^{i-1} - x^*\|^2 - \|x^i - x^*\|^2\right).$$

从 $i = 1$ 到 $i = k$ 累加所有的不等式，并利用 $\psi(x^i)$ 是非增的，可得上界估计

$$\psi(x^k) - \psi^* \leqslant \frac{1}{2kt_{\min}}\|x^0 - x^*\|^2. \qquad \square$$

7.2 Nesterov 加速算法

上一节分析了近似点梯度算法的收敛速度：如果光滑部分的梯度是利普希茨连续的，则目标函数的收敛速度可以达到 $\mathcal{O}\left(\frac{1}{k}\right)$. 一个自然的问题

是如果仅用梯度信息，我们能不能取得更快的收敛速度．Nesterov 分别在 1983 年、1988 年和 2005 年提出了三种改进的一阶算法，收敛速度能达到 $\mathcal{O}\left(\frac{1}{k^2}\right)$. 实际上，这三种算法都可以应用到近似点梯度算法上. 在 Nesterov 加速算法刚提出的时候，由于牛顿算法有更快的收敛速度，Nesterov 加速算法在当时并没有引起太多的关注．但近年来，随着数据量的增大，牛顿型方法由于其过大的计算复杂度，不便于有效地应用到实际中，Nesterov 加速算法作为一种快速的一阶算法重新被挖掘出来并迅速流行起来．Beck 和 Teboulle 就在 2008 年给出了 Nesterov 在 1983 年提出的算法的近似点梯度法版本——FISTA．本节将对这些加速方法做一定的介绍和总结，主要讨论凸函数的加速算法，并给出相应的例子和收敛性证明．作为补充，我们也将简单介绍非凸问题上的加速算法.

7.2.1 FISTA 算法

考虑如下复合优化问题：

$$\min_{x\in\mathbb{R}^n} \quad \psi(x) = f(x) + h(x), \tag{7.2.1}$$

其中 $f(x)$ 是连续可微的凸函数且梯度是利普希茨连续的（利普希茨常数是 L），$h(x)$ 是适当的闭凸函数．优化问题(7.2.1) 由光滑部分 $f(x)$ 和非光滑部分 $h(x)$ 组成，可以使用近似点梯度法来求解这一问题，但是其收敛速度只有 $\mathcal{O}\left(\frac{1}{k}\right)$．很自然地，我们希望能够加速近似点梯度算法，这就是本小节要介绍的 FISTA 算法.

FISTA 算法由两步组成：第一步沿着前两步的计算方向计算一个新点，第二步在该新点处做一步近似点梯度迭代，即

$$\begin{aligned} y^k &= x^{k-1} + \frac{k-2}{k+1}(x^{k-1} - x^{k-2}), \\ x^k &= \operatorname{prox}_{t_k h}(y^k - t_k \nabla f(y^k)). \end{aligned}$$

图7.2给出 FISTA 算法的迭代序列图．可以看到这一做法对每一步迭代的计算量几乎没有影响，而带来的效果是显著的．如果选取 t_k 为固定的步长并小于或等于 $\frac{1}{L}$，其收敛速度达到了 $\mathcal{O}\left(\frac{1}{k^2}\right)$，我们将在收敛性分析中给出推导过程．完整的 FISTA 算法见算法7.2.

为了对算法做更好的推广，可以给出 FISTA 算法的一个等价变形，只是把原来算法中的第一步拆成两步迭代，相应算法见算法 7.3 ．当 $\gamma_k = \frac{2}{k+1}$

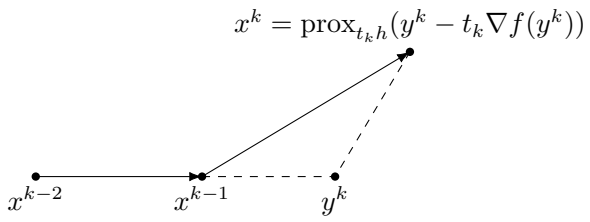


图 7.2 FISTA 一次迭代

算法 7.2 FISTA

1. **输入**：$x^0 = x^{-1} \in \mathbb{R}^n$，$k \leftarrow 1$.
2. **while** 未达到收敛准则 **do**
3. 计算 $y^k = x^{k-1} + \dfrac{k-2}{k+1}(x^{k-1} - x^{k-2})$,
4. 选取 $t_k = t \in \left(0, \dfrac{1}{L}\right]$，计算 $x^k = \mathrm{prox}_{t_k h}(y^k - t_k \nabla f(y^k))$.
5. $k \leftarrow k + 1$.
6. **end while**

时，并且取固定步长时，两个算法是等价的. 但是当 γ_k 采用别的取法时，算法 7.3 将给出另一个版本的加速算法.

算法 7.3 FISTA 算法的等价变形

1. **输入**：$v_0 = x_0 \in \mathbb{R}^n$，$k \leftarrow 1$.
2. **while** 未达到收敛准则 **do**
3. 计算 $y^k = (1 - \gamma_k)x^{k-1} + \gamma_k v^{k-1}$.
4. 选取 t_k，计算 $x^k = \mathrm{prox}_{t_k h}(y^k - t_k \nabla f(y^k))$.
5. 计算 $v^k = x^{k-1} + \dfrac{1}{\gamma_k}(x^k - x^{k-1})$.
6. $k \leftarrow k + 1$.
7. **end while**

对于该算法框架，我们需要确定如何选取步长 t_k 和 γ_k，这决定了算法的收敛速度. 首先给出算法 7.3 以 $\mathcal{O}\left(\dfrac{1}{k^2}\right)$ 的速度收敛的条件（具体的证明

将在后面给出)：

$$f(x^k) \leqslant f(y^k) + \langle \nabla f(y^k), x^k - y^k \rangle + \frac{1}{2t_k}\|x^k - y^k\|_2^2, \tag{7.2.2}$$

$$\gamma_1 = 1, \quad \frac{(1-\gamma_i)t_i}{\gamma_i^2} \leqslant \frac{t_{i-1}}{\gamma_{i-1}^2}, \quad i > 1, \tag{7.2.3}$$

$$\frac{\gamma_k^2}{t_k} = \mathcal{O}\left(\frac{1}{k^2}\right). \tag{7.2.4}$$

可以看到当取 $t_k = \dfrac{1}{L}$，$\gamma_k = \dfrac{2}{k+1}$ 时，以上条件满足．而且 γ_k 的选取并不唯一，例如我们可以采取

$$\gamma_1 = 1 \quad \frac{1}{\gamma_k} = \frac{1}{2}\left(1 + \sqrt{1 + \frac{4}{\gamma_{k-1}}}\right)$$

来得到序列 $\{\gamma_k\}$，这样导出的算法的收敛速度依然是 $\mathcal{O}\left(\dfrac{1}{k^2}\right)$.

在算法 7.2 和算法 7.3 中都要求步长满足 $t_k \leqslant \dfrac{1}{L}$，此时条件(7.2.2)满足．然而，对绝大多数问题我们不知道函数 ∇f 的利普希茨常数．为了在这种情况下条件(7.2.2)依然能满足，需要使用线搜索来确定合适的 t_k，同时选取 γ_k 使得条件(7.2.3)和条件(7.2.4) 同时满足，从而使得算法达到 $\mathcal{O}\left(\dfrac{1}{k^2}\right)$ 的收敛速度．下面给出两个线搜索算法，在执行它们时条件(7.2.2)-(7.2.4)同时得到满足，进而可以得到相同收敛速度的结合线搜索的 FISTA 算法.

第一种方法比较直观，类似于近似点梯度算法，它是在算法 7.3的第 4 行中加入线搜索，并取 $\gamma_k = \dfrac{2}{k+1}$，以回溯的方式找到满足条件 (7.2.2)的 t_k 即可．每一次的起始步长取为前一步的步长 t_{k-1}，通过不断指数式地减小步长 t_k 来使得条件(7.2.2)得到满足．注意，当 t_k 足够小时，条件(7.2.2)是一定会得到满足的，因此不会出现线搜索无法终止的情况．容易验证其他两个条件(7.2.3)(7.2.4) 在迭代过程中也得到满足．该算法的具体过程见算法 7.4.

在第一种方法中线搜索的初始步长 t_k 取为上一步的步长 t_{k-1}，并在迭代过程中不断减小，这不利于算法较快收敛．注意到算法7.3中参数 γ_k 也是可调的，这进一步增加了设计线搜索算法的灵活性．第二种线搜索方法不仅改变步长 t_k 而且改变 γ_k，所以 y^k 也随之改变．该算法的具体过程见 7.5.

由 γ_k 的计算可知，其一定满足条件(7.2.3)且有 $0 < \gamma_k \leqslant 1$，并且 t_k 的选

算法 7.4 线搜索算法 1

1. **输入**：$t_k = t_{k-1} > 0$，$\rho < 1$. 参照点 y^k 及其梯度 $\nabla f(y^k)$.
2. 计算 $x^k = \mathrm{prox}_{t_k h}(y^k - t_k \nabla f(y^k))$.
3. **while** 条件(7.2.2)对 x^k, y^k 不满足 **do**
4. $\quad t_k \leftarrow \rho t_k$.
5. $\quad$ 重新计算 $x^k = \mathrm{prox}_{t_k h}(y^k - t_k \nabla f(y^k))$.
6. **end while**
7. **输出**：迭代点 x^k，步长 t_k.

算法 7.5 线搜索算法 2

1. **输入**：$t_k = \hat{t} > 0$，$t_{k-1} > 0$，$\rho < 1$.
2. **loop**
3. $\quad$ 取 γ_k 为关于 γ 的方程 $t_{k-1}\gamma^2 = t_k \gamma_{k-1}^2 (1-\gamma)$ 的正根.
4. $\quad$ 计算 $y^k = (1-\gamma_k)x^{k-1} + \gamma_k v^{k-1}$ 和梯度 $\nabla f(y^k)$.
5. $\quad$ 计算 $x^k = \mathrm{prox}_{t_k h}(y^k - t_k \nabla f(y^k))$.
6. $\quad$ **if** 条件(7.2.2)对 x^k, y^k 不满足 **then**
7. $\quad\quad t_k \leftarrow \rho t_k$.
8. $\quad$ **else**
9. $\quad\quad$ 结束循环.
10. $\quad$ **end if**
11. **end loop**
12. **输出**：迭代点 x^k，步长 t_k.

取必有一个下界 $t_{\min}$. 关于 $\dfrac{\gamma_k^2}{t_k}$ 的估计，我们有

$$\frac{\sqrt{t_{k-1}}}{\gamma_{k-1}} = \frac{\sqrt{(1-\gamma_k)t_k}}{\gamma_k} \leqslant \frac{\sqrt{t_k}}{\gamma_k} - \frac{\sqrt{t_k}}{2},$$

这里的不等号是由于 $\sqrt{1-x}$ 在点 $x=0$ 处的凹性. 反复利用上式可得

$$\frac{\sqrt{t_k}}{\gamma_k} \geqslant \sqrt{t_1} + \frac{1}{2}\sum_{i=2}^{k}\sqrt{t_i},$$

因此

$$\frac{\gamma_k^2}{t_k} \leqslant \frac{1}{(\sqrt{t_1} + \frac{1}{2}\sum_{i=2}^{k}\sqrt{t_i})^2} \leqslant \frac{4}{t_{min}(k+1)^2} = \mathcal{O}\left(\frac{1}{k^2}\right). \tag{7.2.5}$$

以上的分析说明了条件(7.2.3)和条件(7.2.4)在算法7.5 的执行中也得到满足.

算法7.5的执行过程比算法7.4的复杂. 由于它同时改变了 t_k 和 γ_k，迭代点 x^k 和参照点 y^k 在线搜索的过程中都发生了变化，点 y^k 处的梯度也需要重新计算. 但此算法给我们带来的好处就是步长 t_k 不再单调下降，在迭代后期也可以取较大值，这会进一步加快收敛.

总的来说，固定步长的 FISTA 算法对于步长的选取是较为保守的，为了保证收敛，有时不得不选取一个很小的步长，这使得固定步长的 FISTA 算法收敛较慢. 如果采用线搜索，则在算法执行过程中会有很大机会选择符合条件的较大步长，因此线搜索可能加快算法的收敛，但代价就是每一步迭代的复杂度变高. 在实际的 FISTA 算法中，需要权衡固定步长和线搜索算法的利弊，从而选择针对特定问题的高效算法.

7.2.2 其他加速算法

本小节将给出除 FISTA 算法外的另外两种加速算法，它们分别是 Nesterov 在 1988 年和 2005 年提出的算法的推广版本. 此外，本小节还将简单提一下针对非凸复合优化问题的 Nesterov 加速算法.

对于复合优化问题 (7.2.1)，我们给出第二类 Nesterov 加速算法，见算法 7.6.

算法 7.6 第二类 Nesterov 加速算法

1. **输入：** 令 $x^0 = y^0$，初始化 $k \leftarrow 1$.
2. **while** 未达到收敛准则 **do**
3. 计算 $z^k = (1-\gamma_k)x^{k-1} + \gamma_k y^{k-1}$.
4. 计算 $y^k = \mathrm{prox}_{(t_k/\gamma_k)h}\left(y^{k-1} - \dfrac{t_k}{\gamma_k}\nabla f(z^k)\right)$.
5. 计算 $x^k = (1-\gamma_k)x^{k-1} + \gamma_k y^k$.
6. $k \leftarrow k+1$.
7. **end while**
8. **输出:** x^k.

第二类 Nesterov 加速算法的一步迭代可参考图 7.3. 和经典 FISTA 算法的一个重要区别在于，第二类 Nesterov 加速算法中的三个序列 $\{x^k\}$，$\{y^k\}$ 和 $\{z^k\}$ 都可以保证在定义域内. 而 FISTA 算法中的序列 $\{y^k\}$ 不一定在定义域内. 在第二类 Nesterov 加速算法中，我们同样可以取 $\gamma_k = \dfrac{2}{k+1}$，$t_k = \dfrac{1}{L}$

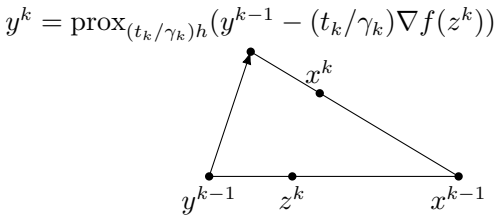


图 7.3 第二类 Nesterov 加速算法的一步迭代

来获得 $\mathcal{O}\left(\frac{1}{k^2}\right)$ 的收敛速度.

针对问题 (7.2.1)的第三类 Nesterov 加速算法框架见算法 7.7. 该算法和

算法 7.7 第三类 Nesterov 加速算法

1. **输入：** 令 $x^0 \in \mathbf{dom}\, h$，$y^0 = \arg\min_{x \in \mathbf{dom}\, h} \|x\|^2$. 初始化 $k \leftarrow 1$.
2. **while** 未达到收敛准则 **do**
3. 计算 $z^k = (1-\gamma_k)x^{k-1} + \gamma_k y^{k-1}$.
4. 计算 $y^k = \mathrm{prox}_{(t_k \sum_{i=1}^k 1/\gamma_i)h}\left(-t_k \sum_{i=1}^k \frac{1}{\gamma_i}\nabla f(z^i)\right)$.
5. 计算 $x^k = (1-\gamma_k)x^{k-1} + \gamma_k y^k$.
6. $k \leftarrow k+1$.
7. **end while**
8. **输出:** x^k.

第二类 Nesterov 加速算法 7.6的区别仅仅在于 y^k 的更新，第三类 Nesterov 加速算法计算 y^k 时需要利用全部已有的 $\{\nabla f(z^i)\}, i = 1,2,\cdots,k$. 同样地，该算法取 $\gamma_k = \frac{2}{k+1}$，$t_k = \frac{1}{L}$ 时，也有 $\mathcal{O}\left(\frac{1}{k^2}\right)$ 的收敛速度.

除了针对凸问题的加速算法，还有针对非凸复合优化问题的加速算法. 仍然考虑问题(7.2.1)的形式，这里并不要求 f 是凸的，但是要求其是可微的且梯度是利普希茨连续的，h 与之前的要求相同. 将针对凸函数的加速算法做一些修改，我们可以给出非凸复合优化问题的加速梯度法框架. 在算法 7.8 中，λ_k 和 t_k 分别为更新 y^k 和 x^k 的步长参数. 从形式上看，算法 7.8 和之前我们接触的任何一种算法都不相同，但可以证明当 λ_k 和 t_k 取特定值时，它等价于之前介绍过的第二类 Nesterov 加速算法（见本章习题）.

算法 7.8 复合优化问题的加速算法框架

1. **输入：** 令 $x^0 = y^0 \in \mathbb{R}^n$，取 $\{\gamma_k\}$ 使得 $\gamma_1 = 1$ 且当 $k \geqslant 2$ 时，$\gamma_k \in (0,1)$. 初始化 $k \leftarrow 1$.
2. **while** 未达到收敛准则 **do**
3. $\quad z^k = \gamma_k y^{k-1} + (1-\gamma_k)x^{k-1}$,
4. $\quad y^k = \mathrm{prox}_{\lambda_k h}\left(y^{k-1} - \lambda_k \nabla f(z^k)\right)$,
5. $\quad x^k = \mathrm{prox}_{t_k h}\left(z^k - t_k \nabla f(z^k)\right)$.
6. $\quad k \leftarrow k+1$.
7. **end while**
8. **输出:** x^k.

在非凸函数情形下，一阶算法一般只能保证收敛到一个稳定点，并不能保证收敛到最优解，因此无法用函数值与最优值的差来衡量优化算法解的精度. 类似于非凸光滑函数利用梯度作为停止准则，对于非凸复合函数 (7.2.1)，我们利用梯度映射（定义 7.2）来判断算法是否收敛. 注意到 $G_t(x) = 0$ 是优化问题(7.2.1)的一阶必要条件，因此利用 $\|G_{t_k}(x^k)\|$ 来刻画算法 7.8 的收敛速度. 可以证明，当 f 为凸函数时，算法 7.8 的收敛速度与 FISTA 算法相同，两者都为 $\mathcal{O}\left(\frac{1}{k^2}\right)$；当 f 为非凸函数时，算法 7.8 也收敛，且收敛速度为 $\mathcal{O}\left(\frac{1}{k}\right)$，详见 [50].

7.2.3 应用举例

之前我们用近似点梯度算法求解的模型，都可以用 Nesterov 加速算法来求解. 为了简化篇幅，此处不再重复叙述对应的优化问题，而是直接给出相应的加速算法迭代格式.

1. LASSO 问题求解

求解 LASSO 问题的 FISTA 算法可以由下面的迭代格式给出：

$$\begin{aligned} y^k &= x^{k-1} + \frac{k-2}{k+1}(x^{k-1} - x^{k-2}), \\ w^k &= y^k - t_k A^T(Ay^k - b), \\ x^k &= \mathrm{sign}(w^k)\max\{|w^k| - t_k\mu, 0\}. \end{aligned}$$

与近似点梯度算法相同，由于最后一步将 w^k 中绝对值小于 $t_k\mu$ 的分量置零，该算法能够保证迭代过程中解具有稀疏结构. 我们也给出第二类 Nesterov 加速算法：

$$\begin{aligned}
z^k &= (1-\gamma_k)x^{k-1} + \gamma_k y^{k-1},\\
w^k &= y^{k-1} - \frac{t_k}{\gamma_k}A^T(Az^k - b),\\
y^k &= \mathrm{sign}(w^k)\max\left\{|w^k| - \frac{t_k}{\gamma_k}\mu, 0\right\},\\
x^k &= (1-\gamma_k)x^{k-1} + \gamma_k y^k,
\end{aligned}$$

和第三类 Nesterov 加速算法：

$$\begin{aligned}
z^k &= (1-\gamma_k)x^{k-1} + \gamma_k y^{k-1},\\
w^k &= -t_k\sum_{i=1}^{k}\frac{1}{\gamma_i}A^T(Az^i - b),\\
y^k &= \mathrm{sign}(w^k)\max\left\{|w^k| - t_k\sum_{i=1}^{k}\frac{1}{\gamma_i}\mu, 0\right\},\\
x^k &= (1-\gamma_k)x^{k-1} + \gamma_k y^k.
\end{aligned}$$

我们用同第5.2节中一样的 A 和 b，并取 $\mu = 10^{-3}$，分别利用连续化近似点梯度法、连续化 FISTA 加速算法、连续化第二类 Nesterov 算法来求解问题，并分别取固定步长 $t = \dfrac{1}{L}$，这里 $L = \lambda_{\max}(A^TA)$，和结合线搜索的 BB 步长. 停机准则与参数 μ 的连续化设置和第5.2节中的光滑化梯度法一致. 结果如图7.4. 可以看到：就固定步长而言，FISTA 算法相较于 第二类 Nesterov 加速算法收敛得略快一些，也可注意到 FISTA 算法是非单调算法. 同时，BB 步长和线搜索技巧可以加速算法的收敛速度. 此外，带线搜索的近似点梯度法可以比带线搜索的 FISTA 算法更快收敛.

7.2.4 收敛性分析

本小节将给出 Nesterov 加速算法收敛速度的理论分析，我们将只针对凸优化问题分析 FISTA 加速算法的收敛速度，而对于第二类和第三类 Nesterov 加速算法和非凸问题的加速算法，有兴趣的读者可以自行阅读相关文献.

下面的定理给出了固定步长的 FISTA 算法的收敛速度.

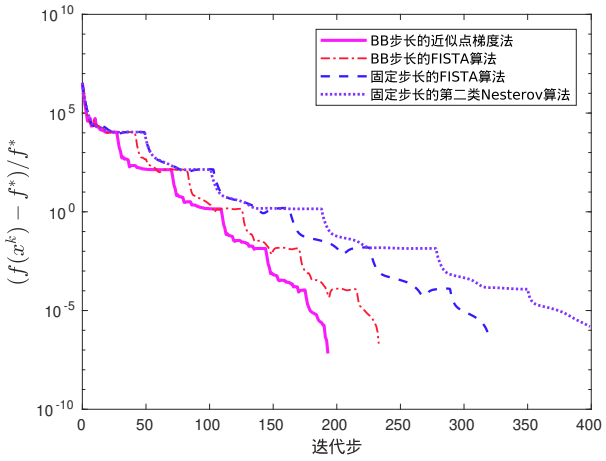


图 7.4 使用近似点梯度法以及不同的加速算法求解 LASSO 问题

定理 7.5 (固定步长 FISTA 算法收敛速度) 在假设 7.1 的条件下，当用算法 7.3 求解凸复合优化问题 (7.2.1) 时，若取固定步长 $t_k = \dfrac{1}{L}$，则

$$\psi(x^k) - \psi(x^*) \leqslant \frac{2L}{(k+1)^2}\|x^0 - x^*\|^2. \tag{7.2.6}$$

证明. 首先根据 $x^k = \mathrm{prox}_{t_k h}(y^k - t_k \nabla f(y^k))$，可知

$$-x^k + y^k - t_k \nabla f(y^k) \in t_k \partial h(x^k).$$

故对于任意的 x，有

$$t_k h(x) \geqslant t_k h(x^k) + \langle -x^k + y^k - t_k \nabla f(y^k), x - x^k \rangle. \tag{7.2.7}$$

另一方面由 f 梯度利普希茨连续和 $t_k = \dfrac{1}{L}$ 可以得到

$$f(x^k) \leqslant f(y^k) + \langle \nabla f(y^k), x^k - y^k \rangle + \frac{1}{2t_k}\|x^k - y^k\|^2. \tag{7.2.8}$$

结合以上两个不等式，对于任意的 x 有

$$\begin{aligned}\psi(x^k) &= f(x^k) + h(x^k) \\ &\leqslant h(x) + f(y^k) + \langle \nabla f(y^k), x - y^k \rangle + \frac{1}{t_k}\langle x^k - y^k, x - x^k \rangle + \\ &\quad \frac{1}{2t_k}\|x^k - y^k\|^2 \\ &\leqslant h(x) + f(x) + \frac{1}{t_k}\langle x^k - y^k, x - x^k \rangle + \frac{1}{2t_k}\|x^k - y^k\|^2 \\ &= \psi(x) + \frac{1}{t_k}\langle x^k - y^k, x - x^k \rangle + \frac{1}{2t_k}\|x^k - y^k\|^2.\end{aligned} \tag{7.2.9}$$

在(7.2.9)式中分别取 $x = x^{k-1}$ 和 $x = x^*$，并记 $\psi(x^*) = \psi^*$，再分别乘 $1 - \gamma_k$ 和 γ_k 并相加得到

$$\begin{aligned}&\psi(x^k) - \psi^* - (1 - \gamma_k)(\psi(x^{k-1}) - \psi^*) \\ \leqslant &\frac{1}{t_k}\langle x^k - y^k, (1 - \gamma_k)x^{k-1} + \gamma_k x^* - x^k \rangle + \frac{1}{2t_k}\|x^k - y^k\|^2.\end{aligned} \tag{7.2.10}$$

结合迭代式

$$\begin{aligned}v^k &= x^{k-1} + \frac{1}{\gamma_k}(x^k - x^{k-1}), \\ y^k &= (1 - \gamma_k)x^{k-1} + \gamma_k v^{k-1},\end{aligned}$$

不等式(7.2.10)可以化为

$$\begin{aligned}&\psi(x^k) - \psi^* - (1 - \gamma_k)(\psi(x^{k-1}) - \psi^*) \\ \leqslant &\frac{1}{2t_k}(\|y^k - (1 - \gamma_k)x^{k-1} - \gamma_k x^*\|^2 - \|x^k - (1 - \gamma_k)x^{k-1} - \gamma_k x^*\|^2) \\ = &\frac{\gamma_k^2}{2t_k}(\|v^{k-1} - x^*\|^2 - \|v^k - x^*\|^2).\end{aligned} \tag{7.2.11}$$

注意到 t_k, γ_k 的取法满足不等式

$$\frac{1 - \gamma_k}{\gamma_k^2}t_k \leqslant \frac{1}{\gamma_{k-1}^2}t_{k-1}, \tag{7.2.12}$$

可以得到一个有关相邻两步迭代的不等式

$$\frac{t_k}{\gamma_k^2}(\psi(x^k) - \psi^*) + \frac{1}{2}\|v^k - x^*\|^2 \leqslant \frac{t_{k-1}}{\gamma_{k-1}^2}(\psi(x^{k-1}) - \psi^*) + \frac{1}{2}\|v^{k-1} - x^*\|^2. \tag{7.2.13}$$

反复利用(7.2.13)式，我们有

$$\frac{t_k}{\gamma_k^2}(\psi(x^k)-\psi^*)+\frac{1}{2}\|v^k-x^*\|^2\leqslant\frac{t_1}{\gamma_1^2}(\psi(x^1)-\psi^*)+\frac{1}{2}\|v^1-x^*\|^2. \quad (7.2.14)$$

对 $k=1$，注意到 $\gamma_1=1, v^0=x^0$，再次利用(7.2.11)式可得

$$\begin{aligned}&\frac{t_1}{\gamma_1^2}(\psi(x^1)-\psi^*)+\frac{1}{2}\|v^1-x^*\|^2\\ \leqslant&\frac{(1-\gamma_1)t_1}{\gamma_1^2}(\psi(x^0)-\psi^*)+\frac{1}{2}\|v^0-x^*\|^2=\frac{1}{2}\|x^0-x^*\|^2.\end{aligned} \quad (7.2.15)$$

结合(7.2.14)式和(7.2.15)式可以得到(7.2.6)式. □

在定理7.5的证明中关键的一步在于建立(7.2.13)式，而建立这个递归关系并不需要 $t=\frac{1}{L},\gamma_k=\frac{2}{k+1}$ 这一具体条件，我们只需要保证条件(7.2.2)和条件(7.2.3)成立即可. 条件(7.2.2)主要依赖于 $f(x)$ 的梯度利普希茨连续性；而(7.2.3) 的成立依赖于 γ_k 和 t_k 的选取. 最后，条件(7.2.4)的成立保证了算法7.3 的收敛速度达到 $\mathcal{O}\left(\frac{1}{k^2}\right)$. 也就是说，如果抽取条件(7.2.2)-(7.2.4) 作为算法收敛的一般条件，则可以证明一大类 FISTA 算法的变形都具有 $\mathcal{O}\left(\frac{1}{k^2}\right)$ 的收敛速度.

推论 7.1 (一般 FISTA 算法的收敛速度) *在假设 7.1的条件下，当用算法 7.3 求解凸复合优化问题 (7.2.1) 时，若迭代点 x^k, y^k，步长 t_k 以及组合系数 γ_k 满足条件(7.2.2)-(7.2.4)，则*

$$\psi(x^k)-\psi(x^*)\leqslant\frac{C}{k^2}, \quad (7.2.16)$$

其中 C 仅与函数 f，初始点 x^0 的选取有关. 特别地，采用线搜索算法7.4和算法7.5的 FISTA 算法具有 $\mathcal{O}\left(\frac{1}{k^2}\right)$ 的收敛速度.

在这里我们指出，虽然已经抽象出了 t_k,γ_k 满足的条件，但我们无法再找到其他的 t_k,γ_k 来进一步改善 FISTA 算法的收敛速度，即 $\mathcal{O}\left(\frac{1}{k^2}\right)$ 是 FISTA 算法所能达到的最高的收敛速度.

7.3 近似点算法

前两节分别讨论了近似点梯度算法和 Nesterov 加速算法，它们能够处理部分不可微的目标函数. 对于一般形式的目标函数，可以用近似点算法，

该算法是近似点梯度算法的一种特殊情况. 我们将其单独拿出来讨论, 是因为近似点算法具有一些特殊的理论性质, 例如其与增广拉格朗日函数法有某种等价关系. 本节首先给出近似点算法的格式和其加速版本, 然后叙述其与增广拉格朗日函数法的关系, 最后讨论近似点算法的收敛性.

7.3.1 近似点算法

本小节将讨论如下一般形式的最优化问题:

$$\min_{x} \quad \psi(x), \tag{7.3.1}$$

其中 ψ 是一个适当的闭凸函数, 这里并不要求 ψ 是可微或连续的 (例如 ψ 的一部分可以是凸集的示性函数). 对于不可微的 ψ, 我们可以用次梯度算法, 但是该方法往往收敛较慢, 且收敛条件比较苛刻. 我们也可以考虑如下隐式格式的次梯度算法:

$$x^{k+1} = x^k - t_k \partial\psi(x^{k+1}). \tag{7.3.2}$$

上面的格式只是形式上的. 类似于之前的近似点梯度算法, 可以用邻近算子表示隐式格式: 近似点算法格式可以写成

$$\begin{aligned} x^{k+1} &= \mathrm{prox}_{t_k\psi}(x^k) \\ &= \arg\min_{u} \left\{ \psi(u) + \frac{1}{2t_k}\|u - x^k\|_2^2 \right\}, \end{aligned} \tag{7.3.3}$$

其中 t_k 为第 k 步迭代时的步长, 可为固定值或通过某种合适的线搜索策略得到. 回顾近似点梯度法的迭代格式, 会发现这个算法可以看做是近似点梯度法在 $f(x) = 0$ 时的情况, 但不同之处在于: 在近似点梯度法中, 非光滑项 $h(x)$ 的邻近算子通常比较容易计算; 而在近似点算法中, $\psi(x)$ 的邻近算子通常是难以求解的, 绝大多数情况下需借助其他类型的迭代法进行 (不精确) 求解.

注 7.1 在近似点算法迭代格式 (7.3.3) 中, 我们构造了一个看似比原问题 (7.3.1) 形式更复杂的子问题. 这样的构造带来的好处是: 子问题的目标函数是一个强凸函数, 更加便于使用迭代法进行求解.

与近似点梯度法类似, 同样可以对近似点算法进行加速. 与其对应的 FISTA 算法的迭代格式可以写成

$$x^k = \mathrm{prox}_{t_k\psi}\left(x^{k-1} + \gamma_k \frac{1-\gamma_{k-1}}{\gamma_{k-1}}(x^{k-1} - x^{k-2}) \right),$$

第二类 Nesterov 加速算法的迭代格式可以写成

$$v^k = \mathrm{prox}_{(t_k/\gamma_k)\psi}(v^{k-1}), \quad x^k = (1-\gamma_k)x^{k-1} + \gamma_k v^k.$$

关于算法参数的选择，我们提出两种策略：

- 策略 1：取固定步长 $t_k = t$ 以及 $\gamma_k = \dfrac{2}{k+1}$；
- 策略 2：对于可变步长 t_k，当 $k = 1$ 时取 $\gamma_1 = 1$；当 $k > 1$ 时，γ_k 由方程

$$\frac{(1-\gamma_k)t_k}{\gamma_k^2} = \frac{t_{k-1}}{\gamma_{k-1}^2}$$

确定.

7.3.2 与增广拉格朗日函数法的关系

这一小节将讨论近似点算法与增广拉格朗日函数法之间的关系. 这是近似点算法一个非常重要的性质，了解该性质能增进对增广拉格朗日函数的理解. 考虑具有如下形式的优化问题：

$$\min_{x\in\mathbb{R}^n} \quad f(x) + h(Ax), \tag{7.3.4}$$

其中 f, h 为适当的闭凸函数. 容易计算出其对偶问题为

$$\max \quad \psi(z) = -f^*(-A^{\mathrm{T}}z) - h^*(z), \tag{7.3.5}$$

其中 f^*, h^* 分别为 f, h 的共轭函数.

问题 (7.3.4) 描述了很广泛的一类凸优化问题，我们给出三个常见的例子.

例 7.4

(1) 当 h 是单点集 $\{b\}$ 的示性函数时，问题(7.3.4)等价于线性等式约束优化问题

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x), \\ \text{s.t.} \quad & Ax = b. \end{aligned}$$

(2) 当 h 是凸集 C 上的示性函数时，问题(7.3.4)等价于约束问题

$$\begin{aligned} \min \quad & f(x), \\ \text{s.t.} \quad & Ax \in C. \end{aligned}$$

(3) 当 $h(y)=\|y-b\|$ 时，问题(7.3.4) 等价于正则优化问题

$$\min \quad f(x)+\|Ax-b\|.$$

对于对偶问题 (7.3.5)，我们使用近似点算法进行更新，即

$$z^{k+1}=\operatorname{prox}_{t\psi}(z^k)=\arg\min_z\left\{f^*(-A^{\mathrm{T}}z)+h^*(z)+\frac{1}{2t_k}\|z-z^k\|_2^2\right\}.$$

而对于原始问题 (7.3.4)，我们引入中间变量 y，可以得到其等价形式

$$\begin{aligned}\min \quad & f(x)+h(y),\\ \text{s.t.} \quad & Ax=y.\end{aligned}\tag{7.3.6}$$

对问题 (7.3.6) 可以用增广拉格朗日函数法进行求解，其迭代格式分为最小化增广拉格朗日函数和对偶更新两步：

$$\begin{aligned}(x^{k+1},y^{k+1})&=\operatorname*{argmin}_{x,y}\left\{f(x)+h(y)+\frac{t_k}{2}\|Ax-y+\frac{1}{t_k}z^k\|_2^2\right\},\\ z^{k+1}&=z^k+t_k(Ax^{k+1}-y^{k+1}).\end{aligned}$$

接下来介绍本节最重要的结论：对对偶问题 (7.3.5) 用近似点算法，实际上等价于对原始问题 (7.3.6) 用增广拉格朗日函数法. 由于问题 (7.3.4) 和问题 (7.3.5) 的自变量不同，我们必须理清二者变量的对应关系. 下面有关共轭函数的结论是证明两个算法等价关系的基础.

命题 7.1 设 $f(x)$ 是适当的闭凸函数，$f^*(y)$ 是其共轭函数，则对任意的 $y\in\mathbf{dom}\,f^*$ 和 $x\in\mathbf{dom}\,f$ 有

$$y\in\partial f(x)\Leftrightarrow x\in\partial f^*(y).$$

证明. 由于 f 是适当闭函数，根据定理 2.12 有 $f^{**}=f$. 根据 $f^*(y)$ 的定义，$y\in\partial f(x)$ 即表明 x 满足最优性条件，因此

$$x^{\mathrm{T}}y-f(x)=f^*(y).$$

由自共轭性得

$$f^{**}(x)=f(x)=x^{\mathrm{T}}y-f^*(y),$$

这说明 y 是最优值点，即 y 满足最优性条件 $x\in\partial f^*(y)$. 另一个方向的结论可类似地得到. □

为了方便，我们将增广拉格朗日函数法一步迭代写为如下形式：

$$(\hat{x},\hat{y}) = \arg\min_{x,y}\left\{f(x)+h(y)+z^{\mathrm{T}}(Ax-y)+\frac{t}{2}\|Ax-y\|_2^2\right\},$$
$$u = z + t(A\hat{x}-\hat{y}).$$

下面证明上述迭代中 u 的更新实际等价于近似点算法的更新.

证明. 问题

$$\min_{x,y} \quad f(x)+h(y)+z^{\mathrm{T}}(Ax-y)+\frac{t}{2}\|Ax-y\|_2^2$$

可以写为

$$\begin{aligned}\min_{x,y,w} \quad & f(x)+h(y)+\frac{t}{2}\|w\|_2^2,\\ \text{s.t.} \quad & Ax-y+\frac{z}{t}=w.\end{aligned}$$

对约束 $Ax-y+\frac{z}{t}=w$ 引入乘子 u，由最优性条件有

$$A\hat{x}-\hat{y}+\frac{z}{t}=w, \quad -A^{\mathrm{T}}u\in\partial f(\hat{x}), \quad u\in\partial h(\hat{y}), \quad tw=u,$$

消去 w 得 $u=z+t(A\hat{x}-\hat{y})$. 下面只需要讨论 $\hat{x}$ 和 $\hat{y}$ 满足的条件. 根据命题 7.1 可知

$$\hat{x}\in\partial f^*(-A^{\mathrm{T}}u), \quad \hat{y}\in\partial h^*(u),$$

代入 u 的表达式最终可得到

$$0\in -A\partial f^*(-A^{\mathrm{T}}u)+\partial h^*(u)+\frac{1}{t}(u-z).$$

这正是 $u=\mathrm{prox}_{t\psi}(z)$ 的最优性条件.

另一方面，若有 $u=\mathrm{prox}_{t\psi}(z)$，则选取 $\hat{x}\in\partial f^*(-A^{\mathrm{T}}u)$ 及 $\hat{y}\in\partial h^*(u)$，即可恢复出增广拉格朗日函数法中的变量. □

本节说明，针对某些形式的问题，对原始问题应用增广拉格朗日函数法等价于对对偶问题应用近似点算法. 实际上，此结论对不少优化问题都是成立的. 由于增广拉格朗日函数法是一类比较有效的处理约束优化问题的算法，根据等价性，如果子问题能够高效求解，近似点算法也应该具有不错的表现. 这一点将在应用举例中具体说明.

7.3.3 应用举例

1. LASSO 问题求解

考虑 LASSO 问题

$$\min_{x\in\mathbb{R}^n} \quad \mu\|x\|_1+\frac{1}{2}\|Ax-b\|_2^2. \tag{7.3.7}$$

引入变量 $y=Ax-b$，问题 (7.3.7)可以等价地转化为

$$\min_{x,y} \quad \psi(x,y)\overset{\text{def}}{=\!=}\mu\|x\|_1+\frac{1}{2}\|y\|_2^2+I_{\mathcal{D}}(x,y), \tag{7.3.8}$$

其中 $\mathcal{D}=\{(x,y)\mid Ax-y=b\}$，$I_{\mathcal{D}}$ 为集合 $\mathcal{D}$ 的示性函数.

对于问题(7.3.8)，我们采用近似点算法进行求解，其第 k 步迭代为

$$(x^{k+1},y^{k+1})\approx\arg\min_{x,y} \quad \left\{\psi(x,y)+\frac{1}{2t_k}\left(\|x-x^k\|_2^2+\|y-y^k\|_2^2\right)\right\}, \tag{7.3.9}$$

其中 t_k 为步长. 由于问题 (7.3.9)没有显式解，我们需要采用迭代算法来进行求解，比如罚函数法、增广拉格朗日函数法等.

除了直接求解问题 (7.3.9)，另一种比较实用的方式是通过对偶问题的解来构造 (x^{k+1},y^{k+1}). 引入拉格朗日乘子 z，问题 (7.3.9)的对偶函数为

$$\begin{aligned}\Phi_k(z)=&\inf_x\left\{\mu\|x\|_1+z^{\mathrm{T}}Ax+\frac{1}{2t_k}\|x-x^k\|_2^2\right\}\\&+\inf_y\left\{\frac{1}{2}\|y\|_2^2-z^{\mathrm{T}}y+\frac{1}{2t_k}\|y-y^k\|_2^2\right\}-b^{\mathrm{T}}z\\=&\mu\Gamma_{\mu t_k}(x^k-t_kA^{\mathrm{T}}z)-\frac{1}{2t_k}\left(\|x^k-t_kA^{\mathrm{T}}z\|_2^2-\|x^k\|_2^2\right)\\&-\frac{t_k}{2(t_k+1)}\|z\|_2^2-\frac{1}{t_k+1}z^{\mathrm{T}}y^k+\frac{1}{2(t_k+1)}\|y^k\|_2^2-b^{\mathrm{T}}z.\end{aligned}$$

这里，

$$\Gamma_{\mu t_k}(u)=\inf_x\left\{\|x\|_1+\frac{1}{2\mu t_k}\|x-u\|_2^2\right\}.$$

注意到 $\Gamma_{\mu t_k}(u)$ 的表达式可以利用 ℓ_1 范数的邻近算子得到，记函数 $q_{\mu t_k}:\mathbb{R}\to\mathbb{R}$ 为

$$q_{\mu t_k}(v)=\begin{cases}\dfrac{v^2}{2\mu t_k}, & |v|\leqslant\mu t_k,\\ |v|-\dfrac{\mu t_k}{2}, & |v|>\mu t_k,\end{cases}$$

则

$$\Gamma_{\mu t_k}(u)=\sum_{i=1}^{n} q_{\mu t_k}(u_i),$$

其为极小点 $x=\mathrm{prox}_{\mu t_k\|\cdot\|_1}(u)$ 处的目标函数值. 易知 $\Gamma_{\mu t_k}(u)$ 是关于 u 的连续可微函数且梯度为

$$\nabla_u \Gamma_{\mu t_k}(u)=\frac{u-\mathrm{prox}_{\mu t_k\|\cdot\|_1}(u)}{\mu t_k}.$$

那么，问题(7.3.9)的对偶问题为

$$\max_{z} \quad \Phi_k(z).$$

设对偶问题的逼近最优解为 z^{k+1}，那么根据问题(7.3.9)的最优性条件，

$$\begin{cases} x^{k+1}=\mathrm{prox}_{\mu t_k\|\cdot\|_1}\left(x^k-t_kA^{\mathrm{T}}z^{k+1}\right), \\ y^{k+1}=\dfrac{1}{t_k+1}(y^k+t_kz^{k+1}). \end{cases}$$

综上，在第 k 步迭代，LASSO 问题 (7.3.7) 的近似点算法的迭代格式写为

$$\begin{cases} z^{k+1}\approx \arg\max\limits_{z} \quad \Phi_k(z), \\ x^{k+1}=\mathrm{prox}_{\mu t_k\|\cdot\|_1}\left(x^k-t_kA^{\mathrm{T}}z^{k+1}\right), \\ y^{k+1}=\dfrac{1}{t_k+1}(y^k+t_kz^{k+1}). \end{cases} \tag{7.3.10}$$

因为 $\Phi_k(z)$ 的最大值点的显式表达式是未知的，所以需要使用迭代算法近似求解. 根据 $\Phi_k(z)$ 的连续可微性，我们可以调用梯度法进行求解. 另外，还可以证明 $\Phi_k(z)$ 是半光滑的，从而调用半光滑牛顿法来更有效地求解，相关内容可以参考 [78]. 为了保证算法(7.3.10)的收敛性，根据文章 [111]，我们采用以下不精确收敛准则：

$$\|\nabla\Phi_k(z^{k+1})\|_2\leqslant\sqrt{\frac{\alpha_k}{t_k}}\varepsilon_k,\quad \varepsilon_k\geqslant 0,\sum_{k=1}^{\infty}\varepsilon_k<\infty,$$

$$\|\nabla\Phi_k(z^{k+1})\|_2\leqslant\sqrt{\frac{\alpha_k}{t_k}}\delta_k\|(x^{k+1},y^{k+1})-(x^k,y^k)\|_2,\quad \delta_k\geqslant 0,\sum_{k=1}^{\infty}\delta_k<+\infty,$$

其中 ε_k,δ_k 是人为设定的参数，α_k 为 Φ_k 的强凹参数（即 $-\Phi_k$ 的强凸参数）的一个估计.

我们用同第5.2节中一样的 A 和 b，分别取 $\mu = 10^{-2}, 10^{-3}$，利用近似点算法来求解问题，取固定步长 $t_k \xlongequal{\text{def}} t = 10^3$，其中子问题利用梯度下降法进行不精确求解，精度参数设置为 $\alpha_k = \dfrac{t}{t+1}$，$\varepsilon_k = \delta_k = \dfrac{8}{k^2}$，结果如图 7.5所示．可以看到：近似点算法收敛所需要的外部迭代数很少，主要计算都在内

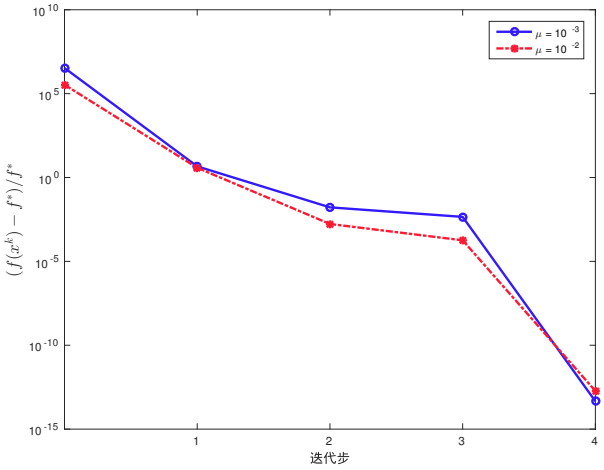


图 7.5 近似点算法求解 LASSO 问题

迭代求解更新 z 的子问题上．对于 z 的子问题求解，我们利用了半光滑牛顿法来进行加速．

7.3.4 收敛性分析

这一小节给出近似点算法的收敛性分析．首先我们给出近似点算法收敛的基本定理．

定理 7.6 *设 ψ 是适当的闭凸函数（从而 $\operatorname{prox}_{t\psi}(x)$ 对任意的 x 存在且唯一），最优值 ψ^* 有限且在点 x^* 处可达，则对近似点算法有*

$$\psi(x^k) - \psi^* \leqslant \frac{\|x^0 - x^*\|_2^2}{2\sum_{i=1}^{k} t_i}, \quad \forall\, k \geqslant 1.$$

证明. 由于近似点算法可看做近似点梯度法的特殊情况，我们的证明也将沿用近似点梯度法中的分析过程，这里仅仅指出关键步骤的不同之处．由于

$f(x)=0$，所以对任意的 $t>0$，

$$f(x-tG_t(x)) \leqslant t\nabla f(x)^{\mathrm{T}}G_t(x)+\frac{t}{2}\|G_t(x)\|_2^2,$$

其中 $G_t(x)$ 的定义见 (7.1.6) 式．运用近似点梯度法中的证明，我们有

$$t_i(\psi(x^i)-\psi^*) \leqslant \frac{1}{2}(\|x^{i-1}-x^*\|_2^2-\|x^i-x^*\|_2^2),$$

且 $\{\psi(x^i)\}$ 是单调下降的序列．因此

$$\left(\sum_{i=1}^{k}t_i\right)(\psi(x^k)-\psi^*) \leqslant \sum_{i=1}^{k}t_i(\psi(x^i)-\psi^*) \leqslant \frac{1}{2}\|x^0-x^*\|_2^2.$$

这样就完成了对定理7.6 的证明． □

上述定理使得我们可以通过每次迭代的步长来控制算法的收敛性．若 $\sum_{i=1}^{k}t_i \to +\infty$，则算法收敛．特别地，如果步长 t_i 固定或有正下界，则收敛速度为 $\mathcal{O}\left(\frac{1}{k}\right)$．与定理 7.3 不同，由于 $f(x)=0$，我们无需对步长 t_i 提出额外的要求，即每一步的 t_i 可以为任意值．理论上可以取充分大的 t_i 使得近似点算法经过较少的迭代步即可收敛，但这种做法实际的意义并不大．这是因为过大的 t_i 会导致子问题难以求解，从近似点算法的形式也可以看出，当 $t_i=+\infty$ 时子问题与原问题是等价的．

类似地，可以给出加速版本的收敛性．

定理 7.7 设 ψ 是适当的闭凸函数，最优值 ψ^* 有限且在点 x^* 处达到．假设参数 t_k,γ_k 按照第 7.3.1 小节中的策略 1 或者策略 2 选取，那么迭代序列 $\{\psi(x^k)\}$ 满足

$$\psi(x^k)-\psi^* \leqslant \frac{2\|x^0-x^*\|_2^2}{\left(2\sqrt{t_1}+\sum\limits_{i=2}^{k}\sqrt{t_i}\right)^2}, \quad k\geqslant 1.$$

证明． 在 $f(x)=0$ 的情况下使用 Nesterov 加速算法中的证明，这里仅指出关键步骤的不同之处．由于 $f(x)=0$, 对任意的 $t>0$，

$$f(x) \leqslant f(y)+\nabla f(y)^{\mathrm{T}}(x-y)+\frac{1}{2t}\|x-y\|_2^2, \quad \forall\, x,y,$$

于是结论

$$\psi(x^k) - \psi^* \leqslant \frac{\gamma_k^2}{2t_k}\|x^0 - x^*\|_2^2 \tag{7.3.11}$$

依然成立. 对于固定步长 $t_k = t$ 和 $\gamma_k = \dfrac{2}{k+1}$,

$$\frac{\gamma_k^2}{2t_k} = \frac{2}{(k+1)^2 t};$$

而对于变步长，根据 (7.2.5) 式，

$$\frac{\gamma_k^2}{2t_k} \leqslant \frac{2}{\left(2\sqrt{t_1} + \sum\limits_{i=2}^{k}\sqrt{t_i}\right)^2}.$$

对于以上两种参数选择策略，分别将对应不等式代入 (7.3.11) 式就得到定理中的结论. □

这个定理意味着当 $\sum\limits_{i=1}^{k}\sqrt{t_i} \to +\infty$ 时算法收敛，且步长 t_i 取固定值或有正下界时，其收敛速度可达到 $\mathcal{O}\left(\dfrac{1}{k^2}\right)$. 同样地，即使理论上可以取 t_i 为任意大的值，实际上仍需控制其上界以便子问题可快速求解.

7.4 分块坐标下降法

在许多实际的优化问题中，人们所考虑的目标函数虽然有成千上万的自变量，对这些变量联合求解目标函数的极小值通常很困难，但这些自变量具有某种“可分离”的形式：当固定其中若干变量时，函数的结构会得到极大的简化. 这种特殊的形式使得人们可以将原问题拆分成数个只有少数自变量的子问题. 分块坐标下降法（block coordinate descent，BCD）正是利用了这样的思想来求解这种具有特殊结构的优化问题，在多数实际问题中有良好的数值表现. 本节介绍分块坐标下降法的基本迭代格式，同时给出一些例子来说明其在具体问题上的应用.

7.4.1 问题描述

考虑具有如下形式的问题：

$$\min_{x\in\mathcal{X}} \quad F(x_1, x_2, \cdots, x_s) = f(x_1, x_2, \cdots, x_s) + \sum_{i=1}^{s} r_i(x_i), \tag{7.4.1}$$

其中 $\mathcal{X}$ 是函数的可行域，这里将自变量 x 拆分成 s 个变量块 $x_1, x_2, \cdots, x_s$，每个变量块 $x_i \in \mathbb{R}^{n_i}$. 函数 f 是关于 x 的可微函数，每个 $r_i(x_i)$ 关于 x_i 是适当的闭凸函数，但不一定可微.

在问题 (7.4.1) 中，目标函数 F 的性质体现在 f，每个 r_i 以及自变量的分块上. 通常情况下，f 对于所有变量块 x_i 不可分，但单独考虑每一块自变量时，f 有简单结构；r_i 只和第 i 个自变量块有关，因此 r_i 在目标函数中是一个可分项. 求解问题 (7.4.1) 的难点在于如何利用分块结构处理不可分的函数 f.

注 7.2 在给出问题 (7.4.1)时，唯一引入凸性的部分是 r_i. 其余部分没有引入凸性，可行域 $\mathcal{X}$ 不是一定是凸集，f 也不一定是凸函数.

需要指出的是，并非所有问题都适合按照问题 (7.4.1) 进行处理. 下面给出六个可以化成问题 (7.4.1) 的实际例子，第 7.4.3 小节将介绍如何使用分块坐标下降法求解它们.

例 7.5 (分组 LASSO[122]) 考虑线性模型 $b = a^{\mathrm{T}}x + \varepsilon$，现在对 x 使用分组 *LASSO* 模型建模. 设矩阵 $A \in \mathbb{R}^{n\times p}$ 和向量 $b \in \mathbb{R}^n$ 分别由上述模型中自变量和响应变量的 n 组观测值组成. 参数 $x = (x_1, x_2, \cdots, x_G) \in \mathbb{R}^p$ 可以分成 G 组，且 $\{x_i\}_{i=1}^G$ 中只有少数的非零向量. 则分组 *LASSO* 对应的优化问题可表示为

$$\min_x \quad \frac{1}{2n}\|b - Ax\|_2^2 + \lambda \sum_{i=1}^{G} \sqrt{p_i}\|x_i\|_2.$$

在这个例子中待优化的变量共有 G 块.

例 7.6 (低秩矩阵恢复 [108]) 设 $b \in \mathbb{R}^m$ 是已知的观测向量，$\mathcal{A}$ 是线性映射. 考虑求解下面的极小化问题：

$$\min_{X,Y} \quad \frac{1}{2}\|\mathcal{A}(XY) - b\|_2^2 + \alpha\|X\|_F^2 + \beta\|Y\|_F^2,$$

其中 $\alpha, \beta > 0$ 为正则化参数. 这里正则化的作用是消除解 (X, Y) 在放缩意义下的不唯一性. 在这个例子中自变量共有两块.

类似的例子还有非负矩阵分解.

例 7.7 (非负矩阵分解 [101]) 设 M 是已知矩阵，考虑求解如下极小化问题：

$$\min_{X,Y\geqslant 0} \quad \frac{1}{2}\|XY - M\|_F^2 + \alpha r_1(X) + \beta r_2(Y).$$

在这个例子中自变量共有两块，且均有非负的约束.

上述的所有例子中，函数 f 关于变量全体一般是非凸的，这使得求解问题(7.4.1)变得很有挑战性. 首先，应用在非凸问题上的算法的收敛性不易分析，很多针对凸问题设计的算法通常会失效；其次，目标函数的整体结构十分复杂，这使得变量的更新需要很大计算量. 对于这类问题，我们最终的目标是要设计一种算法，它具有简单的变量更新格式，同时具有一定的（全局）收敛性. 而分块坐标下降法则是处理这类问题较为有效的算法.

7.4.2 算法结构

考虑问题 (7.4.1)，我们所感兴趣的分块坐标下降法具有如下更新方式：按照 $x_1, x_2, \cdots, x_s$ 的次序依次固定其他 $(s-1)$ 块变量极小化 F，完成一块变量的极小化后，它的值便立即被更新到变量空间中，更新下一块变量时将使用每个变量最新的值. 根据这种更新方式定义辅助函数

$$f_i^k(x_i) = f(x_1^k, \cdots, x_{i-1}^k, x_i, x_{i+1}^{k-1}, \cdots, x_s^{k-1}),$$

其中 x_j^k 表示在第 k 次迭代中第 j 块自变量的值，x_i 是函数的自变量. 函数 f_i^k 表示在第 k 次迭代更新第 i 块变量时所需要考虑的目标函数的光滑部分. 考虑第 i 块变量时前 $(i-1)$ 块变量已经完成更新，因此上标为 k，而后面下标从 $(i+1)$ 起的变量仍为旧的值，因此上标为 $(k-1)$.

在每一步更新中，通常使用以下三种更新格式之一：

$$x_i^k = \arg\min_{x_i \in \mathcal{X}_i^k} \left\{ f_i^k(x_i) + r_i(x_i) \right\}, \tag{7.4.2}$$

$$x_i^k = \arg\min_{x_i \in \mathcal{X}_i^k} \left\{ f_i^k(x_i) + \frac{L_i^{k-1}}{2} \|x_i - x_i^{k-1}\|_2^2 + r_i(x_i) \right\}, \tag{7.4.3}$$

$$x_i^k = \arg\min_{x_i \in \mathcal{X}_i^k} \left\{ \langle \hat{g}_i^k, x_i - \hat{x}_i^{k-1} \rangle + \frac{L_i^{k-1}}{2} \|x_i - \hat{x}_i^{k-1}\|_2^2 + r_i(x_i) \right\}, \tag{7.4.4}$$

其中 $L_i^k > 0$ 为常数，

$$\mathcal{X}_i^k = \{ x \in \mathbb{R}^{n_i} \mid (x_1^k, \cdots, x_{i-1}^k, x, x_{i+1}^{k-1}, \cdots, x_s^{k-1}) \in \mathcal{X} \}.$$

在更新格式(7.4.4)中，$\hat{x}_i^{k-1}$ 采用外推定义：

$$\hat{x}_i^{k-1} = x_i^{k-1} + \omega_i^{k-1}(x_i^{k-1} - x_i^{k-2}), \tag{7.4.5}$$

其中 $\omega_i^k \geqslant 0$ 为外推的**权重**，$\hat{g}_i^k \overset{\text{def}}{=\!=} \nabla f_i^k(\hat{x}_i^{k-1})$ 为外推点处的梯度. 在(7.4.5)式中取权重 $\omega_i^k = 0$ 即可得到不带外推的更新格式，此时计算(7.4.4)等价于进行

一次近似点梯度法的更新. 在(7.4.4)式使用外推是为了加快分块坐标下降法的收敛速度. 我们可以通过如下的方式理解这三种格式: 格式(7.4.2)是最直接的, 即固定其他分量然后对单一变量求极小; 格式(7.4.3)则是增加了一个近似点项 $\frac{L_i^{k-1}}{2}\|x_i - x_i^{k-1}\|_2^2$ 来限制下一步迭代不应该与当前位置相距过远, 增加近似点项的作用是使得算法能够收敛; 格式(7.4.4) 首先对 $f_i^k(x)$ 进行线性化以简化子问题的求解, 在此基础上引入了 Nesterov 加速算法的技巧加快收敛.

为了直观地说明分块坐标下降法的迭代过程, 我们给出一个简单的例子.

例 7.8 考虑二元二次函数的优化问题

$$\min \quad f(x,y) = x^2 - 2xy + 10y^2 - 4x - 20y,$$

现在对变量 x,y 使用分块坐标下降法求解. 当固定 y 时, 可知当 $x = 2 + y$ 时函数取极小值; 当固定 x 时, 可知当 $y = 1 + \frac{x}{10}$ 时函数取极小值. 故采用格式(7.4.2)的分块坐标下降法为

$$x^{k+1} = 2 + y^k, \tag{7.4.6}$$

$$y^{k+1} = 1 + \frac{x^{k+1}}{10}. \tag{7.4.7}$$

图 7.6描绘了当初始点为 $(x,y) = (0.5,0.2)$ 时的迭代点轨迹, 可以看到在进行了 7 次迭代后迭代点与最优解已充分接近. 回忆一下我们在例 5.2 中曾经对一个类似的问题使用过梯度法, 而梯度法的收敛相当缓慢. 一个直观的解释是: 对于比较病态的问题, 由于分块坐标下降法是对逐个分量处理, 它能较好地捕捉目标函数的各向异性, 而梯度法则会受到很大影响.

结合上述更新格式(7.4.2)—(7.4.4) 可以得到分块坐标下降法的基本框架, 详见算法 7.9.

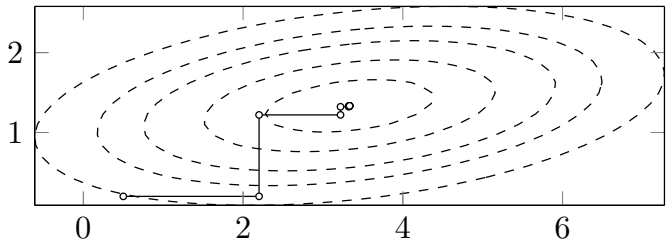


图 7.6 分块坐标下降法迭代轨迹

算法 7.9 分块坐标下降法

1. **初始化：** 选择两组初始点 $(x_1^{-1}, x_2^{-1}, \cdots, x_s^{-1}) = (x_1^0, x_2^0, \cdots, x_s^0)$.
2. **for** $k = 1, 2, \cdots$ **do**
3. **for** $i = 1, 2, \cdots$ **do**
4. 使用格式 (7.4.2) 或 (7.4.3) 或 (7.4.4) 更新 x_i^k.
5. **end for**
6. **if** 满足停机条件 **then**
7. 返回 $(x_1^k, x_2^k, \cdots, x_s^k)$，算法终止.
8. **end if**
9. **end for**

算法7.9的子问题可采用三种不同的更新格式，一般来说这三种格式会产生不同的迭代序列，可能会收敛到不同的解，坐标下降算法的数值表现也不相同. 格式(7.4.2)是最直接的更新方式，它严格保证了整个迭代过程的目标函数值是下降的. 然而由于 f 的形式复杂，子问题求解难度较大. 在收敛性方面，格式(7.4.2)在强凸问题上可保证目标函数收敛到极小值，但在非凸问题上不一定收敛. 格式(7.4.3) (7.4.4) 则是对格式(7.4.2)的修正，不保证迭代过程目标函数的单调性，但可以改善收敛性结果. 使用格式(7.4.3)可使得算法收敛性在函数 F 为非严格凸时有所改善. 格式(7.4.4)实质上为目标函数的一阶泰勒展开近似，在一些测试问题上有更好的表现，可能的原因是使用一阶近似可以避开一些局部极小值点. 此外，格式(7.4.4)的计算量很小，比较容易实现.

在实际的应用中，三种更新格式都有适用的问题，如果子问题可以写出

显式解，则使用分块坐标下降算法可以节省相当一部分计算量. 在每一步更新中，三种迭代格式(7.4.2) — (7.4.4) 对不同自变量块可以混合使用，不必仅仅局限于一种. 但对于同一个变量块，在整个迭代中应该使用相同的格式. 例如在之后介绍的字典学习问题中，若对变量 D 使用格式(7.4.2)，对变量 X 使用格式(7.4.4)，则两个子问题都有显式解. 因此更新格式的混用使得分块坐标下降法变得更加灵活.

7.4.3 应用举例

1. LASSO 问题求解

下面介绍如何使用分块坐标下降法来求解 LASSO 问题

$$\min_x \mu\|x\|_1 + \frac{1}{2}\|Ax - b\|^2. \tag{7.4.8}$$

由于目标函数的 $\|x\|_1$ 部分是可分的，因此第 i 块变量即为 x 的第 i 个分量. 为了方便，在考虑第 i 块的更新时，将自变量 x 记为

$$x = \begin{bmatrix} x_i \\ \bar{x}_i \end{bmatrix}$$

其中 $\bar{x}_i$ 为 x 去掉第 i 个分量而形成的列向量. 而相应地，矩阵 A 在第 i 块的更新记为

$$A = \begin{bmatrix} a_i & \bar{A}_i \end{bmatrix},$$

其中 $\bar{A}_i$ 为矩阵 A 去掉第 i 列而形成的矩阵. 这里为了方便表示，同时将 x 和 A 的分量顺序进行了调整，但调整后的问题依然和原问题是等价的.

以下我们推导分块坐标下降法的更新格式. 在第 i 块的更新中，考虑直接极小化的格式(7.4.2)，原问题可以写为

$$\min_{x_i} \quad \mu|x_i| + \mu\|\bar{x}_i\|_1 + \frac{1}{2}\|a_i x_i - (b - \bar{A}_i\bar{x}_i)\|^2. \tag{7.4.9}$$

做替换 $c_i = b - \bar{A}_i\bar{x}_i$，并注意到仅与 $\bar{x}_i$ 有关的项是常数，原问题等价于

$$\min_{x_i} \quad f_i(x_i) \overset{\text{def}}{=\!=} \mu|x_i| + \frac{1}{2}\|a_i\|^2 x_i^2 - a_i^{\mathrm{T}} c_i x_i. \tag{7.4.10}$$

对函数(7.4.10)，可直接写出它的最小值点

$$x_i^k = \arg\min_{x_i} f_i(x_i) = \begin{cases} \dfrac{a_i^{\mathrm{T}} c_i - \mu_i}{\|a_i\|^2}, & a_i^{\mathrm{T}} c_i > \mu, \\ \dfrac{a_i^{\mathrm{T}} c_i + \mu_i}{\|a_i\|^2}, & a_i^{\mathrm{T}} c_i < -\mu, \\ 0, & \text{其他}. \end{cases} \tag{7.4.11}$$

因此可写出 LASSO 问题的分块坐标下降法，见算法 7.10.

算法 7.10 LASSO 问题的分块坐标下降法

1. 输入 A, b，参数 μ. 初始化 $x^0 = 0$，$k \leftarrow 1$.
2. **while** 未达到收敛准则 **do**
3. **for** $i = 1, 2, \cdots, n$ **do**
4. 根据定义计算 $\bar{x}_i, c_i$.
5. 使用公式(7.4.11)计算 x_i^k.
6. **end for**
7. $k \leftarrow k + 1$.
8. **end while**

我们用同第5.2节中一样的 A 和 b，分别取 $\mu = 10^{-2}, 10^{-3}$，并调用连续化坐标下降法进行求解，其中停机准则和参数 μ 的连续化设置和第5.2节中的光滑化梯度法一致，结果如图7.7所示. 可以看到，在结合连续化策略之后，坐标下降法可以很快地收敛到问题的解. 相比其他算法，坐标下降法不需要调节步长参数.

2. 非负矩阵分解

非负矩阵分解问题 [101] 也可以使用分块坐标下降法求解. 现在考虑最基本的非负矩阵分解问题

$$\min_{X,Y \geqslant 0} \quad \frac{1}{2}\|XY - M\|_F^2. \tag{7.4.12}$$

它的一个等价形式为

$$\min_{X,Y} \quad \frac{1}{2}\|XY - M\|_F^2 + I_{\geqslant 0}(X) + I_{\geqslant 0}(Y), \tag{7.4.13}$$

其中 $I_{\geqslant 0}(\cdot)$ 为集合 $\{X \mid X \geqslant 0\}$ 的示性函数. 不难验证问题 (7.4.13) 具有形式(7.4.1).

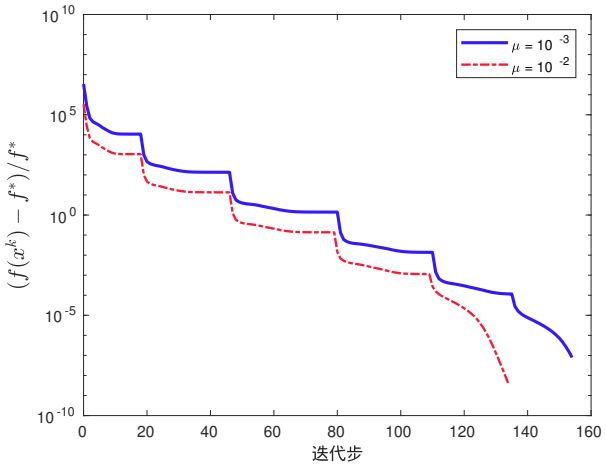


图 7.7 分块坐标下降法求解 LASSO 问题

以下考虑求解方法. 注意到 X 和 Y 耦合在一起，在固定 Y 的条件下，我们无法直接按照格式(7.4.2) 或格式(7.4.3)的形式给出子问题的显式解. 若要采用这两种格式需要额外设计算法求解子问题，最终会产生较大计算量. 但我们总能使用格式(7.4.4)来对子问题进行线性化，从而获得比较简单的更新格式. 令 $f(X,Y)=\frac{1}{2}\|XY-M\|_F^2$，则

$$\frac{\partial f}{\partial X}=(XY-M)Y^{\mathrm{T}},\quad \frac{\partial f}{\partial Y}=X^{\mathrm{T}}(XY-M). \tag{7.4.14}$$

注意到在格式(7.4.4)中，当 $r_i(X)$ 为凸集示性函数时即是求解到该集合的投影，因此得到分块坐标下降法如下：

$$\begin{aligned} X^{k+1}&=\max\{X^k-t_k^x(X^kY^k-M)(Y^k)^{\mathrm{T}},0\},\\ Y^{k+1}&=\max\{Y^k-t_k^y(X^k)^{\mathrm{T}}(X^kY^k-M),0\}, \end{aligned} \tag{7.4.15}$$

其中 t_k^x,t_k^y 是步长，分别对应格式(7.4.4)中的 $\frac{1}{L_i^k},i=1,2$.

7.5 对偶算法

对于复合优化问题，许多实际问题的原始问题有时候比较难以处理，这时候一个非常重要的技巧是考虑它的对偶问题. 本节将讲解两种算法：一种是把前面提到的算法应用到对偶问题上，例如对偶近似点梯度法；另一种是

同时把原始问题和对偶问题结合起来考虑，例如原始 – 对偶混合梯度类的算法. 我们将看到这两种算法的思想将极大地丰富求解问题的手段. 为了方便起见，这一节主要考虑如下形式的问题：

$$\text{(P)}\quad \min_{x\in\mathbb{R}^n}\quad \psi(x)=f(x)+h(Ax), \tag{7.5.1}$$

其中 f,h 都是闭凸函数，$A\in\mathbb{R}^{m\times n}$ 为实数矩阵. 通过引入约束 $y=Ax$，可以写出与问题(7.5.1)等价的约束优化问题：

$$\begin{aligned}\min\quad & f(x)+h(y),\\ \text{s.t.}\quad & y=Ax.\end{aligned} \tag{7.5.2}$$

对约束 $y=Ax$ 引入乘子 z，得到拉格朗日函数

$$\begin{aligned}L(x,y,z)&=f(x)+h(y)-z^{\mathrm{T}}(y-Ax)\\ &=\big(f(x)+(A^{\mathrm{T}}z)^{\mathrm{T}}x\big)+\big(h(y)-z^{\mathrm{T}}y\big).\end{aligned}$$

利用共轭函数的定义(2.6.1)，可计算拉格朗日对偶问题为

$$\text{(D)}\quad \max_{z}\quad \phi(z)=-f^*(-A^{\mathrm{T}}z)-h^*(z). \tag{7.5.3}$$

特别值得注意的是，本章讲解的算法不局限于求解上述形式的问题，在学习过程中注意灵活推广到求解其他形式的问题.

7.5.1 对偶近似点梯度法

在实际应用中，我们会发现许多时候对偶问题的求解会比原始问题的求解容易很多，这时候可以把之前讲过的各种算法，例如梯度下降算法、近似点梯度算法、增广拉格朗日函数法等，应用到对偶问题上. 本小节我们将近似点梯度算法应用到对偶问题上，得到对偶近似点梯度法，还将讨论与其等价的、针对原始问题设计的算法.

对偶问题(7.5.3)是无约束的复合优化形式，因此可以考虑第 7.1 节的近似点梯度算法. 要使用该算法，首先要解决的问题是对偶问题的目标函数 $\phi(z)$ 是否是“可微函数 + 凸函数”的复合形式. 如果假设原始问题中 $f(x)$ 是闭的强凸函数（强凸参数为 μ），下面的引理说明其共轭函数是定义在全空间 $\mathbb{R}^n$ 上的梯度利普希茨连续函数：

引理 7.1 (强凸函数共轭函数的性质) *设 $f(x)$ 是适当且闭的强凸函数，其强凸参数为 $\mu>0$，$f^*(y)$ 是 $f(x)$ 的共轭函数，则 $f^*(y)$ 在全空间 $\mathbb{R}^n$ 上有定义，且 $f^*(y)$ 是梯度 $\dfrac{1}{\mu}$-利普希茨连续的可微函数.*

引理 7.1 指出 ∇f^* 是利普希茨连续函数，因此在对偶问题(7.5.3)中 $f^*(-A^{\mathrm{T}}z)$ 是梯度 $\dfrac{1}{\mu}\|A\|_2^2$-利普希茨连续的函数，这是因为对于任意的 z_1, z_2，有

$$\begin{aligned}\|A\nabla f^*(-A^{\mathrm{T}}z_1) - A\nabla f^*(-A^{\mathrm{T}}z_2)\| &\leqslant \frac{1}{\mu}\|A\|_2\|A^{\mathrm{T}}(z_1 - z_2)\| \\ &\leqslant \frac{\|A\|_2^2}{\mu}\|z_1 - z_2\|.\end{aligned}$$

考虑在对偶问题上应用近似点梯度算法，每次迭代更新如下：

$$z^{k+1} = \mathrm{prox}_{th^*}(z^k + tA\nabla f^*(-A^{\mathrm{T}}z^k)), \tag{7.5.4}$$

这里注意对偶问题是取最大值，因此邻近算子内部应该取上升方向. 引入变量 $x^{k+1} = \nabla f^*(-A^{\mathrm{T}}z^k)$，利用共轭函数的性质得 $-A^{\mathrm{T}}z^k \in \partial f(x^{k+1})$. 因此迭代格式(7.5.4)等价于

$$\begin{aligned}x^{k+1} &= \arg\min_x\{f(x) + (A^{\mathrm{T}}z^k)^{\mathrm{T}}x\}, \\ z^{k+1} &= \mathrm{prox}_{th^*}(z^k + tAx^{k+1}).\end{aligned} \tag{7.5.5}$$

迭代格式(7.5.5)仅仅将计算 $\nabla f^*(-A^{\mathrm{T}}z^k)$ 化成了一个共轭函数的求解问题，本质上和迭代格式(7.5.4)是一样的. 但下面的分析会提供另一种角度来理解对偶近似点梯度法.

我们先引入有关邻近算子和共轭函数的一个重要性质：Moreau 分解.

引理 7.2 (Moreau 分解) *设 f 是定义在 $\mathbb{R}^n$ 上的适当的闭凸函数，则对任意的 $x \in \mathbb{R}^n$，*

$$x = \mathrm{prox}_f(x) + \mathrm{prox}_{f^*}(x); \tag{7.5.6}$$

或更一般地，

$$x = \mathrm{prox}_{\lambda f}(x) + \lambda\,\mathrm{prox}_{\lambda^{-1}f^*}\left(\frac{x}{\lambda}\right), \tag{7.5.7}$$

其中 $\lambda > 0$ 为任意正实数.

Moreau 分解的结论非常漂亮，它表明：对任意的闭凸函数 f，空间 $\mathbb{R}^n$ 上的恒等映射总可以分解成两个函数 f 与 f^* 邻近算子的和. 根据 Moreau 分解的一般形式（取 $\lambda = t, f = h^*$，并注意到 $h^{**} = h$），我们有

$$\begin{aligned}z^k + tAx^{k+1} &= \mathrm{prox}_{th^*}(z^k + tAx^{k+1}) + t\,\mathrm{prox}_{t^{-1}h}\left(\frac{z^k}{t} + Ax^{k+1}\right) \\ &= z^{k+1} + t\,\mathrm{prox}_{t^{-1}h}(\frac{z^k}{t} + Ax^{k+1}),\end{aligned}$$

由此给出对偶近似点梯度法等价的针对原始问题的更新格式：

$$\begin{aligned} x^{k+1} &= \arg\min_x \left\{ f(x) + (z^k)^{\mathrm{T}} A x \right\}, \\ y^{k+1} &= \mathrm{prox}_{t^{-1}h}\left(\frac{z^k}{t} + A x^{k+1} \right) \\ &= \arg\min_y \left\{ h(y) - (z^k)^{\mathrm{T}}(y - A x^{k+1}) + \frac{t}{2}\| A x^{k+1} - y \|_2^2 \right\}, \\ z^{k+1} &= z^k + t(A x^{k+1} - y^{k+1}). \end{aligned} \tag{7.5.8}$$

如果我们写出约束优化问题(7.5.2)的拉格朗日函数和增广拉格朗日函数

$$\begin{aligned} L(x,y,z) &= f(x) + h(y) - z^{\mathrm{T}}(y - Ax), \\ L_t(x,y,z) &= f(x) + h(y) - z^{\mathrm{T}}(y - Ax) + \frac{t}{2}\| y - Ax \|^2, \end{aligned}$$

则迭代格式(7.5.8)可以等价地写为

$$\begin{aligned} x^{k+1} &= \arg\min_x L(x, y^k, z^k), \\ y^{k+1} &= \arg\min_y L_t(x^{k+1}, y, z^k), \\ z^{k+1} &= z^k + t(A x^{k+1} - y^{k+1}). \end{aligned} \tag{7.5.9}$$

迭代格式(7.5.9)又称为**交替极小化方法**：第一步迭代为在拉格朗日函数中关于 x 求极小，第二步迭代为在增广拉格朗日函数中关于 y 求极小，第三步迭代为更新拉格朗日乘子. 交替极小化方法与下一节介绍的交替方向乘子法有非常相似的结构. 上面的分析表明，对偶近似点梯度法等价于对原始问题(7.5.2)使用交替极小化方法. 若 f 可分，可将 x 的求解划分成几个独立的子问题求解；在 z 的更新中，步长 t 可以为常数，或者由线搜索决定. 在上述更新框架下，还可以考虑引入加速版本的近似点梯度算法.

下面我们给出四个例子来说明如何拆分和使用对偶近似点梯度法.

例 7.9 (正则化范数近似) 假设 f 是强凸函数，考虑

$$\min\ f(x) + \| Ax - b \|,$$

其中 $\|\cdot\|$ 是任意一种范数. 对应原始问题(7.5.1)我们有 $h(y) = \|y - b\|$，可计算出 $h(y)$ 的共轭函数为

$$h^*(z) = \begin{cases} b^{\mathrm{T}} z, & \|z\|_* \leqslant 1 \\ +\infty, & \text{其他}, \end{cases}$$

其中 $\|\cdot\|_*$ 表示 $\|\cdot\|$ 的对偶范数. 从而对偶问题为:

$$\max_{\|z\|_*\leqslant 1} \quad -f^*(-A^{\mathrm{T}}z) - b^{\mathrm{T}}z,$$

应用对偶近似点梯度法, 更新如下:

$$x^{k+1} = \arg\min_x \left\{f(x) + (A^{\mathrm{T}}z^k)^{\mathrm{T}}x\right\},$$
$$z^{k+1} = \mathcal{P}_{\|z\|_*\leqslant 1}(z^k + t(Ax^{k+1} - b)).$$

例 7.10 假设 f 是强凸函数, 考虑

$$\min \quad f(x) + \sum_{i=1}^{p}\|B_i x\|_2,$$

即 $h(y_1, y_2, \cdots, y_p) = \sum_{i=1}^{p}\|y_i\|_2$, 且

$$A = \begin{bmatrix} B_1^{\mathrm{T}} & B_1^{\mathrm{T}} & \cdots & B_p^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}.$$

假定 $B_i \in \mathbb{R}^{m_i\times n}$, 记 C_i 是 $\mathbb{R}_{m_i}$ 中的单位欧几里得球, 根据 $\|\cdot\|_2$ 的共轭函数定义, 对偶问题形式如下:

$$\max_{\|z_i\|_2\leqslant 1} \quad -f^*\left(-\sum_{i=1}^{p}B_i^{\mathrm{T}}z_i\right),$$

从而对偶近似点梯度法更新如下:

$$x^{k+1} = \arg\min_x \left\{f(x) + (\sum_{i=1}^{p}B_i^{\mathrm{T}}z_i)^{\mathrm{T}}x\right\},$$
$$z_i^{k+1} = \mathcal{P}_{C_i}(z_i^k + tB_i x^{k+1}), i = 1, 2, \cdots, p.$$

例 7.11 (在凸集交上的极小化) 假设 f 是强凸函数, 考虑

$$\begin{aligned} \min \quad & f(x), \\ \text{s.t.} \quad & x \in C_1 \cap C_2 \cap \cdots \cap C_m, \end{aligned}$$

其中 C_i 为闭凸集, 易于计算投影. 记 $h(y_1, y_2, \cdots, y_m) = \sum_{i=1}^{m}I_{C_i}(y_i)$, 以及

$$A = \begin{bmatrix} I & I & \cdots & I \end{bmatrix}^{\mathrm{T}},$$

从而对偶问题形式如下：

$$\max_{z_i \in C_i} \quad -f^*\left(-\sum_{i=1}^{m} z_i\right) - \sum_{i=1}^{m} I_{C_i}^*(z_i),$$

利用共轭函数的性质可知 $I_{C_i}^*(z_i)$ 是集合 C_i 的支撑函数，其显式表达式不易求出．因此我们利用 *Moreau* 分解将迭代格式写成交替极小化方法的形式：

$$\begin{aligned}
x^{k+1} &= \arg\min_x \left\{ f(x) + \left(\sum_{i=1}^{m} z_i\right)^{\mathrm{T}} x \right\}, \\
y_i^{k+1} &= \mathcal{P}_{C_i}\left(\frac{z_i^k}{t} + x^{k+1}\right), \quad i = 1,2,\cdots,m, \\
z_i^{k+1} &= z_i^k + t(x^{k+1} - y_i^{k+1}), \quad i = 1,2,\cdots,m.
\end{aligned}$$

例 7.12 (可分问题的拆分) 假设 f_i 是强凸函数，h_i^* 有易于计算的邻近算子．考虑

$$\min \quad \sum_{j=1}^{n} f_j(x_j) + \sum_{i=1}^{m} h_i(A_{i1}x_1 + A_{i2}x_2 + \cdots + A_{in}\overline{x}_N),$$

其对偶问题形式如下：

$$\max \quad -\sum_{i=1}^{m} h_i^*(z_i) - \sum_{j=1}^{n} f_j^*(-A_{1j}^{\mathrm{T}} z_1 - A_{2j}^{\mathrm{T}} z_2 - \cdots - A_{mj}^{\mathrm{T}} z_m).$$

对偶近似点梯度法更新如下：

$$\begin{aligned}
x_j^{k+1} &= \arg\min_{x_j} \left\{ f_j(x_j) + (\sum_{i=1}^{m} A_{ij} z_i^k)^{\mathrm{T}} x_j \right\}, \quad j = 1,2,\cdots,n, \\
z_i^{k+1} &= \mathrm{prox}_{th_i^*}\left(z_i + t\sum_{j=1}^{n} A_{ij} x_j^{k+1}\right), \quad i = 1,2,\cdots,m.
\end{aligned}$$

7.5.2 原始 – 对偶混合梯度算法

本小节介绍另外一种非常重要的算法——原始 – 对偶混合梯度（primal-dual hybrid gradient, PDHG）算法．对于给定的优化问题，我们可以从原始问题或对偶问题出发来设计相应算法求解. 那么能否将两个方面结合起来呢？PDHG 算法就是借鉴了这样的思想．相比于直接在原始问题上或者对偶问题上应用优化算法求解，PDHG 算法在每次迭代时同时考虑原始变量

和对偶变量的更新. 这使得它在一定程度上可以有效避免单独针对原始问题或对偶问题求解算法中可能出现的问题，例如，原始问题梯度为零向量或不可微，对偶问题形式复杂等. 本小节将介绍原始 – 对偶混合梯度算法以及它的一个变形——Chambolle-Pock 算法.

PDHG 算法的构造要从鞍点问题谈起. 我们仍然考虑原始问题(7.5.1)：

$$\min \quad f(x)+h(Ax),$$

其中 f,h 是适当的闭凸函数. 由于 h 有自共轭性，我们将问题(7.5.1)变形为

$$(\mathrm{L_{PD}}) \quad \min_x \max_z \quad \psi_{PD}(x,z) \overset{\text{def}}{=\!=} f(x)-h^*(z)+z^{\mathrm{T}}Ax. \tag{7.5.10}$$

可以看到此时问题(7.5.1)变成了一个极小 – 极大问题，即关于变量 x 求极小，关于变量 z 求极大，这是一个典型的鞍点问题.

另一种常用的鞍点问题定义方式是直接利用带约束的问题(7.5.2)构造拉格朗日函数. 问题

$$\min_{x\in\mathbb{R}^n,y\in\mathbb{R}^m} \quad f(x)+h(y), \quad \text{s.t.} \quad y=Ax.$$

相应的鞍点问题形式如下：

$$(\mathrm{L_P}) \quad \min_{x,y} \max_z \quad f(x)+h(y)+z^{\mathrm{T}}(Ax-y). \tag{7.5.11}$$

类似地，在对偶问题(7.5.3)中引入变量 $w=-A^{\mathrm{T}}z$，则可定义鞍点问题

$$(\mathrm{L_D}) \quad \min_p \max_{w,z} \quad -f^*(w)-h^*(z)+p^{\mathrm{T}}(w+A^{\mathrm{T}}z). \tag{7.5.12}$$

鞍点问题是关于某些变量求极小的同时关于另一些变量求极大，直接求解比较困难. PDHG 算法的思想就是分别对两类变量应用近似点梯度算法. 以求解问题 (7.5.10) 为例，PDHG 算法交替更新原始变量以及对偶变量，其迭代格式如下：

$$\begin{aligned} z^{k+1} &= \arg\max_z \left\{ -h^*(z)+\left\langle Ax^k, z-z^k\right\rangle - \frac{1}{2\delta_k}\|z-z^k\|_2^2 \right\} \\ &= \mathrm{prox}_{\delta_k h^*}(z^k+\delta_k Ax^k), \\ x^{k+1} &= \arg\min_x \left\{ f(x)+(z^{k+1})^{\mathrm{T}}A(x-x^k)+\frac{1}{2\alpha_k}\|x-x^k\|_2^2 \right\} \\ &= \mathrm{prox}_{\alpha_k f}(x^k-\alpha_k A^{\mathrm{T}}z^{k+1}), \end{aligned} \tag{7.5.13}$$

其中 α_k,δ_k 分别为原始变量和对偶变量的更新步长. 它在第一步固定原始变量 x^k 针对对偶变量做梯度上升，在第二步固定更新后的对偶变量 z^{k+1} 针对原始变量做梯度下降. 在这里注意，原始变量和对偶变量的更新顺序是无关紧要的，若先更新原始变量，其等价于在另一初值下先更新对偶变量.

事实上，PDHG 算法在 $\alpha_k=+\infty,\delta_k=+\infty$ 这两种特殊情况下等价于近似点梯度算法分别应用于对偶问题和原始问题. 首先考虑 $\alpha_k=+\infty$ 的情形. 此时交换两步迭代的顺序，PDHG 算法(7.5.13)可以写为

$$\begin{aligned} x^{k+1} &= \arg\min_x \{f(x)+(z^k)^{\mathrm{T}}Ax\}, \\ z^{k+1} &= \operatorname{prox}_{\delta_k h^*}(z^k+\delta_k Ax^{k+1}). \end{aligned} \tag{7.5.14}$$

可以看到，其实质上就是迭代格式(7.5.5)，其中 δ_k 是步长. 类似地，我们也可以写出 $\delta_k=+\infty$ 的情形：

$$\begin{aligned} z^{k+1} &= \arg\min_z \{h^*(z)-(Ax^k)^{\mathrm{T}}z\}, \\ x^{k+1} &= \operatorname{prox}_{\alpha_k f}(x^k-\alpha_k A^{\mathrm{T}}z^{k+1})). \end{aligned} \tag{7.5.15}$$

假定 h 和 h^* 都是可微的，由迭代格式(7.5.15)的第一式的最优性条件，可知 $Ax^k=\nabla h^*(z^{k+1})$. 再利用共轭函数的性质，有 $z^{k+1}=\nabla h(Ax^k)$，所以 x^{k+1} 的更新等价于

$$x^{k+1}=\operatorname{prox}_{\alpha_k f}(x^k-\alpha_k A^{\mathrm{T}}\nabla h(Ax^k)).$$

这实际上就是对原始问题应用近似点梯度算法.

PDHG 算法的收敛性需要比较强的条件，在有些情形下未必收敛. 这里再介绍 PDHG 算法的一个变形——Chambolle-Pock 算法 [28]. 它与 PDHG 算法的区别在于多了一个外推步，具体的迭代格式如下：

$$\begin{aligned} z^{k+1} &= \operatorname{prox}_{\delta_k h^*}(z^k+\delta_k Ay^k), \\ x^{k+1} &= \operatorname{prox}_{\alpha_k f}(x^k-\alpha_k A^{\mathrm{T}}z^{k+1}), \\ y^{k+1} &= 2x^{k+1}-x^k. \end{aligned} \tag{7.5.16}$$

我们将不加证明地指出，当取常数步长 $\alpha_k=t,\delta_k=s$ 时，该算法的收敛性在 $\sqrt{st}<\dfrac{1}{\|A\|_2}$ 的条件下成立.

7.5.3 应用举例

1. LASSO 问题求解

考虑 LASSO 问题

$$\min_{x\in\mathbb{R}^n} \quad \psi(x) \xlongequal{\text{def}} \mu\|x\|_1 + \frac{1}{2}\|Ax-b\|_2^2.$$

我们取 $f(x)=\mu\|x\|_1$ 和 $h(x)=\dfrac{1}{2}\|x-b\|_2^2$，相应的鞍点问题(7.5.10)形式如下：

$$\min_{x\in\mathbb{R}^n}\max_{z\in\mathbb{R}^m} \quad f(x)-h^*(z)+z^{\mathrm{T}}Ax.$$

根据共轭函数的定义，

$$h^*(z)=\sup_{y\in\mathbb{R}^m}\left\{y^{\mathrm{T}}z-\frac{1}{2}\|y-b\|_2^2\right\}=\frac{1}{2}\|z\|_2^2+b^{\mathrm{T}}z.$$

应用 PDHG 算法, x^{k+1} 和 z^{k+1} 的更新格式分别为

$$\begin{aligned}
z^{k+1} &= \operatorname{prox}_{\delta_k h^*}(z^k+\delta_k Ax^k)=\frac{1}{\delta_k+1}\left(z^k+\delta_k Ax^k-\delta_k b\right),\\
x^{k+1} &= \operatorname{prox}_{\alpha_k\mu\|\cdot\|_1}(x^k-\alpha_k A^{\mathrm{T}}z^{k+1}).
\end{aligned}$$

这里 δ_k,α_k 为步长. 同样地，可以写出 Chambolle-Pock 算法格式为

$$\begin{aligned}
z^{k+1} &= \frac{1}{\delta_k+1}\left(z^k+\delta_k Ay^k-\delta_k b\right),\\
x^{k+1} &= \operatorname{prox}_{\alpha_k\mu\|\cdot\|_1}(x^k-\alpha_k A^{\mathrm{T}}z^{k+1}),\\
y^{k+1} &= 2x^{k+1}-x^k.
\end{aligned}$$

我们用同第5.2节中一样的 A 和 b，并取 $\mu=10^{-3}$，分别采用带连续化策略的 PDHG 算法和 Chambolle-Pock 算法来进行求解，这里取 $\delta_k=1$ 和 $\alpha_k=\dfrac{1}{\|A\|_2^2}$. 停机准则和参数 μ 的连续化设置与第5.2节中的光滑化梯度法一致. 结果如图7.8所示. 可以看到，尽管 PDHG 算法在某些情况下没有收敛性保证，但它在这个例子上比 Chambolle-Pock 算法稍快一些.

2. TV-L^1 模型

考虑去噪情形下的 TV-L^1 模型:

$$\min_{U\in\mathbb{R}^{n\times n}} \quad \|U\|_{TV}+\lambda\|U-B\|_1,$$

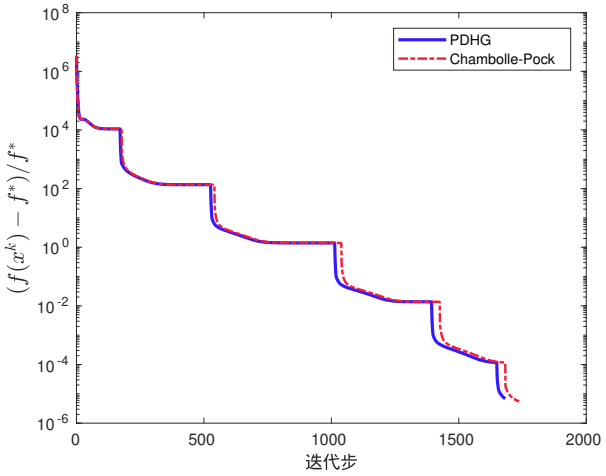


图 7.8 PDHG 算法和 Chambolle-Pock 算法求解 LASSO 问题

其中 $\|U\|_{TV}$ 为全变差，即可以用离散的梯度（线性）算子 $D:\mathbb{R}^{n\times n}\to\mathbb{R}^{n\times n\times 2}$ 表示为

$$\|U\|_{TV}=\sum_{1\leqslant i,j\leqslant n}\|(DU)_{ij}\|_2.$$

对任意的 $W,V\in\mathbb{R}^{n\times n\times 2}$，记

$$\|W\|=\sum_{1\leqslant i,j\leqslant n}\|w_{ij}\|_2,\quad \langle W,V\rangle=\sum_{1\leqslant i,j\leqslant n,1\leqslant k\leqslant 2}w_{i,j,k}v_{i,j,k},$$

其中 $w_{ij}\in\mathbb{R}^2$ 且 $\|\cdot\|$ 定义了 $\mathbb{R}^{n\times n\times 2}$ 上的一种范数. 利用 $\|\cdot\|$ 的定义，有

$$\|U\|_{TV}=\|DU\|.$$

对应于问题(7.5.1)，我们取 D 为相应的线性算子，并取

$$f(U)=\lambda\|U-B\|_1,U\in\mathbb{R}^{n\times n},\quad h(W)=\|W\|,\quad W\in\mathbb{R}^{n\times n\times 2}.$$

相应的鞍点问题 (7.5.10) 如下：

$$(\mathrm{L_{PD}})\quad \min_{U\in\mathbb{R}^{n\times n}}\max_{V\in\mathbb{R}^{n\times n\times 2}}f(U)-h^*(V)+\langle V,DU\rangle.$$

根据共轭函数的定义，

$$h^*(V)=\sup_{U\in\mathbb{R}^{n\times n\times 2}}\{\langle U,V\rangle-\|U\|\}=\begin{cases}0, & \max\limits_{i,j}\|v_{ij}\|_2\leqslant 1,\\ +\infty, & \text{其他.}\end{cases}$$

记 $\mathcal{V}=\{V\in\mathbb{R}^{n\times n\times 2}:\max\limits_{ij}\|v_{ij}\|_2\leqslant 1\}$，其示性函数记为 $I_{\mathcal{V}}(V)$，则问题 $(\mathrm{L_{PD}})$ 可以整理为

$$\min_U\max_V\quad f(U)+\langle V,DU\rangle-I_{\mathcal{V}}(V).$$

应用 PDHG 算法，则 V^{k+1} 的更新为

$$V^{k+1}=\mathrm{prox}_{sI_{\mathcal{V}}}(V^k+sDU^k)=\mathcal{P}_{\mathcal{V}}(V^k+sDU^k),\tag{7.5.17}$$

即 V^k+sDU^k 在 $\mathcal{V}$ 上的投影，而 U^{k+1} 的更新如下：

$$\begin{aligned}U^{k+1}&=\mathrm{prox}_{tf}(U^k+tGV^{k+1})\\&=\arg\min_U\left\{\lambda\|U-B\|_1+\left\langle V^{k+1},DU\right\rangle+\frac{1}{2t}\|U-U^k\|_F^2\right\}\\\Longleftrightarrow(U^{k+1})_{ij}&=\begin{cases}(U^k+tGV^{k+1})_{ij}-t\lambda, & (U^k+tGV^{k+1})_{ij}>B_{ij}+t\lambda,\\(U^k+tGV^{k+1})_{ij}+t\lambda, & (U^k+tGV^{k+1})_{ij}<B_{ij}-t\lambda,\\B_{ij}, & |(U^k+tGV^{k+1})_{ij}-B_{ij}|\leqslant t\lambda,\end{cases}\end{aligned}$$

其中 $G:\mathbb{R}^{n\times n\times 2}\to\mathbb{R}^{n\times n}$ 为离散的散度算子，其满足

$$\langle V,DU\rangle=-\langle GV,U\rangle,\quad\forall\,U\in\mathbb{R}^{n\times n},\,V\in\mathbb{R}^{n\times n\times 2}.$$

若应用 Chambolle-Pock 算法，那么 U^{k+1} 的更新保持不变，仅需调整 V^{k+1} 的更新为 $V^k+sD(2U^{k+1}-U^k)$ 在 $\mathcal{V}$ 上的投影.

7.6 交替方向乘子法

统计学、机器学习和科学计算中出现了很多结构复杂且可能非凸、非光滑的优化问题. 交替方向乘子法很自然地提供了一个适用范围广泛、容易理解和实现、可靠性不错的解决方案. 该方法是在 20 世纪 70 年代发展起来的，与许多其他算法等价或密切相关，如对偶分解、乘子方法、Douglas-Rachford Splitting 方法、Dykstra 交替投影方法、Bregman 对于带 ℓ_1 范数问题的迭代算法、近似点算法等. 本节首先介绍交替方向乘子法的基本算法；在介绍了 Douglas-Rachford Splitting 方法之后，说明将其应用在对偶问题上与将交替方向乘子法应用在原始问题上等价；然后给出交替方向乘子法的一些变形技巧，以及它和其他一些算法的关系；接着给出大量实际问题中的例子，并展示如何用交替方向乘子法来求解这些问题.

7.6.1 交替方向乘子法

本节考虑如下凸问题：

$$\begin{aligned} \min_{x_1,x_2} \quad & f_1(x_1)+f_2(x_2), \\ \text{s.t.} \quad & A_1x_1+A_2x_2=b, \end{aligned} \tag{7.6.1}$$

其中 f_1, f_2 是适当的闭凸函数，但不要求是光滑的，$x_1 \in \mathbb{R}^n, x_2 \in \mathbb{R}^m$, $A_1 \in \mathbb{R}^{p\times n}, A_2 \in \mathbb{R}^{p\times m}$, $b \in \mathbb{R}^p$. 这个问题的特点是目标函数可以分成彼此分离的两块，但是变量被线性约束结合在一起. 常见的一些无约束和带约束的优化问题都可以表示成这一形式. 下面的一些例子将展示如何把某些一般的优化问题转化为适用交替方向乘子法求解的标准形式.

例 7.13 可以分成两块的无约束优化问题

$$\min_x \quad f_1(x)+f_2(x).$$

为了将此问题转化为标准形式(7.6.1)，需要将目标函数改成可分的形式. 我们可以通过引入一个新的变量 z 并令 $x=z$，将问题转化为

$$\begin{aligned} \min_{x,z} \quad & f_1(x)+f_2(z), \\ \text{s.t.} \quad & x-z=0. \end{aligned}$$

例 7.14 带线性变换的无约束优化问题

$$\min_x \quad f_1(x)+f_2(Ax).$$

类似地，我们可以引入一个新的变量 z，令 $z=Ax$，则问题变为

$$\begin{aligned} \min_{x,z} \quad & f_1(x)+f_2(z), \\ \text{s.t.} \quad & Ax-z=0. \end{aligned}$$

对比问题(7.6.1)可知 $A_1=A$ 和 $A_2=-I$.

例 7.15 凸集上的约束优化问题

$$\begin{aligned} \min_x \quad & f(x), \\ \text{s.t.} \quad & Ax \in C, \end{aligned}$$

其中 $C \subset \mathbb{R}^n$ 为凸集. 对于集合约束 $Ax \in C$，我们可以用示性函数 $I_C(\cdot)$ 将其添加到目标函数中，那么问题可以转化为例 7.14 中的形式：

$$\min_x \quad f(x) + I_C(Ax),$$

其中 $I_C(z)$ 是集合 C 的示性函数，即

$$I_C(z) = \begin{cases} 0, & z \in C, \\ +\infty, & \text{其他}. \end{cases}$$

再引入约束 $z = Ax$，那么问题转化为

$$\begin{aligned} \min_{x,z} \quad & f(x) + I_C(z), \\ \text{s.t.} \quad & Ax - z = 0. \end{aligned}$$

例 7.16 全局一致性问题（global consensus problem）*[19]*

$$\min_x \quad \sum_{i=1}^N \phi_i(x).$$

令 $x = z$，并将 x 复制 N 份，分别为 x_i，那么问题转化为

$$\begin{aligned} \min_{x_i,z} \quad & \sum_{i=1}^N \phi_i(x_i), \\ \text{s.t.} \quad & x_i - z = 0,\ i = 1,2,\cdots,N. \end{aligned}$$

在这里注意，从形式上看全局一致性问题仍然具有问题(7.6.1)的结构：如果令

$$x = (x_1^{\mathrm{T}}, x_2^{\mathrm{T}}, \cdots, x_N^{\mathrm{T}})^{\mathrm{T}}$$

以及

$$f_1(x) = \sum_{i=1}^N \phi_i(x_i), \quad f_2(z) = 0,$$

则此问题可以化为

$$\begin{aligned} \min_{x,z} \quad & f_1(x) + f_2(z), \\ \text{s.t.} \quad & A_1 x - A_2 z = 0, \end{aligned}$$

其中矩阵 A_1, A_2 定义为：

$$A_1 = \begin{bmatrix} I & & \\ & I & \\ & & \ddots \\ & & & I \end{bmatrix}, \quad A_2 = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix}.$$

例 7.17 共享问题（Sharing Problem）*[19]*

$$\min_{x_i} \quad \sum_{i=1}^{N} f_i(x_i) + g\left(\sum_{i=1}^{N} x_i\right).$$

为了使目标函数可分，我们将 g 的变量 x_i 分别复制一份为 z_i，那么问题转化为

$$\begin{aligned} \min_{x_i, z_i} \quad & \sum_{i=1}^{N} f_i(x_i) + g\left(\sum_{i=1}^{N} z_i\right), \\ \text{s.t.} \quad & x_i - z_i = 0,\ i = 1, 2, \cdots, N. \end{aligned}$$

容易验证此问题也具有问题(7.6.1)的形式.

下面给出交替方向乘子法（alternating direction method of multipliers, ADMM）的迭代格式，首先写出问题(7.6.1)的增广拉格朗日函数

$$\begin{aligned} L_\rho(x_1, x_2, y) =& f_1(x_1) + f_2(x_2) + y^{\mathrm{T}}(A_1 x_1 + A_2 x_2 - b) \\ &+ \frac{\rho}{2}\|A_1 x_1 + A_2 x_2 - b\|_2^2, \end{aligned} \tag{7.6.2}$$

其中 $\rho > 0$ 是二次罚项的系数. 常见的求解带约束问题的增广拉格朗日函数法为如下更新：

$$(x_1^{k+1}, x_2^{k+1}) = \arg\min_{x_1, x_2} L_\rho(x_1, x_2, y^k), \tag{7.6.3}$$

$$y^{k+1} = y^k + \tau\rho(A_1 x_1^{k+1} + A_2 x_2^{k+1} - b), \tag{7.6.4}$$

其中 τ 为步长. 在实际求解中，第一步迭代(7.6.3)同时对 x_1 和 x_2 进行优化有时候比较困难，而固定一个变量求解关于另一个变量的极小问题可能比较简单，因此我们可以考虑对 x_1 和 x_2 交替求极小，这就是交替方向乘子法的基本思路. 其迭代格式可以总结如下：

$$x_1^{k+1} = \arg\min_{x_1} L_\rho(x_1, x_2^k, y^k), \tag{7.6.5}$$

$$x_2^{k+1} = \arg\min_{x_2} L_\rho(x_1^{k+1}, x_2, y^k), \tag{7.6.6}$$

$$y^{k+1} = y^k + \tau\rho(A_1 x_1^{k+1} + A_2 x_2^{k+1} - b), \tag{7.6.7}$$

其中 τ 为步长，通常取值于 $\left(0, \dfrac{1+\sqrt{5}}{2}\right]$.

观察交替方向乘子法的迭代格式，第一步固定 x_2, y 对 x_1 求极小；第二步固定 x_1, y 对 x_2 求极小；第三步更新拉格朗日乘子 y. 这一迭代格式和之前讨论的交替极小化方法 (7.5.9) 非常相似. 它们的区别是交替极小化方法的第一步是针对拉格朗日函数求极小，而 ADMM 的第一步将其换成了增广拉格朗日函数. 虽然从形式上看两个算法只是略有差别，但这种改变会带来截然不同的算法表现. ADMM 的一个最直接的改善就是去掉了目标函数 $f_1(x)$ 强凸的要求，其本质还是由于它引入了二次罚项. 而在交替极小化方法中我们要求 $f(x)$ 为强凸函数.

需要注意的是，虽然交替方向乘子法引入了二次罚项，但对一般的闭凸函数 f_1 和 f_2，迭代(7.6.5)和迭代 (7.6.6)在某些特殊情况下仍然不是良定义的. 本节假设每个子问题的解均是存在且唯一的，但读者应当注意到这个假设对一般的闭凸函数是不成立的.

与无约束优化问题不同，交替方向乘子法针对的问题(7.6.1)是带约束的优化问题，因此算法的收敛准则应当借助约束优化问题的最优性条件（KKT 条件）. 因为 f_1, f_2 均为闭凸函数，约束为线性约束，所以当 Slater 条件成立时，可以使用凸优化问题的 KKT 条件来作为交替方向乘子法的收敛准则. 问题(7.6.1)的拉格朗日函数为

$$L(x_1, x_2, y) = f_1(x_1) + f_2(x_2) + y^{\mathrm{T}}(A_1x_1 + A_2x_2 - b).$$

根据定理 4.11，若 x_1^*, x_2^* 为问题(7.6.1)的最优解，y^* 为对应的拉格朗日乘子，则以下条件满足：

$$0 \in \partial_{x_1} L(x_1^*, x_2^*, y^*) = \partial f_1(x_1^*) + A_1^{\mathrm{T}} y^*, \tag{7.6.8a}$$

$$0 \in \partial_{x_2} L(x_1^*, x_2^*, y^*) = \partial f_2(x_2^*) + A_2^{\mathrm{T}} y^*, \tag{7.6.8b}$$

$$A_1x_1^* + A_2x_2^* = b. \tag{7.6.8c}$$

在这里条件(7.6.8c)又称为原始可行性条件，条件(7.6.8a)和条件(7.6.8b)又称为对偶可行性条件. 由于问题中只含等式约束，KKT 条件中的互补松弛条件可以不加考虑. 在 ADMM 迭代中，我们得到的迭代点实际为 (x_1^k, x_2^k, y^k)，因此收敛准则应当针对 (x_1^k, x_2^k, y^k) 检测条件(7.6.8). 接下来讨论如何具体计算这些收敛准则.

一般来说，原始可行性条件(7.6.8c)在迭代中是不满足的，为了检测这个条件，需要计算原始可行性残差

$$r^k = A_1x_1^k + A_2x_2^k - b$$

的模长，这一计算是比较容易的.下面来看两个对偶可行性条件.考虑 ADMM 迭代更新 x_2 的步骤

$$x_2^k = \arg\min_x \left\{ f_2(x) + \frac{\rho}{2} \left\| A_1 x_1^k + A_2 x - b + \frac{y^{k-1}}{\rho} \right\|^2 \right\},$$

假设这一子问题有显式解或能够精确求解，根据最优性条件不难推出

$$0 \in \partial f_2(x_2^k) + A_2^{\mathrm{T}}[y^{k-1} + \rho(A_1 x_1^k + A_2 x_2^k - b)]. \tag{7.6.9}$$

注意到当 ADMM 步长 $\tau = 1$ 时，根据迭代(7.6.7)可知上式方括号中的表达式就是 y^k，最终我们有

$$0 \in \partial f_2(x_2^k) + A_2^{\mathrm{T}} y^k,$$

这恰好就是条件(7.6.8b). 上面的分析说明在 ADMM 迭代过程中，若 x_2 的更新能取到精确解且步长 $\tau = 1$，对偶可行性条件(7.6.8b)是自然成立的，因此无需针对条件(7.6.8b)单独验证最优性条件. 然而，在迭代过程中条件(7.6.8a)却不能自然满足. 实际上，由 x_1 的更新公式

$$x_1^k = \arg\min_x \left\{ f_1(x) + \frac{\rho}{2} \| A_1 x + A_2 x_2^{k-1} - b + \frac{y^{k-1}}{\rho} \|^2 \right\},$$

假设子问题能精确求解，根据最优性条件

$$0 \in \partial f_1(x_1^k) + A_1^{\mathrm{T}}[\rho(A_1 x_1^k + A_2 x_2^{k-1} - b) + y^{k-1}].$$

注意，这里 x_2 上标是 $k-1$，因此根据 ADMM 的第三式 (7.6.7)，同样取 $\tau = 1$，我们有

$$0 \in \partial f_1(x_1^k) + A_1^{\mathrm{T}}(y^k + \rho A_2(x_2^{k-1} - x_2^k)). \tag{7.6.10}$$

对比条件 (7.6.8a) 可知多出来的项为 $\rho A_1^{\mathrm{T}} A_2(x_2^{k-1} - x_2^k)$，因此要检测对偶可行性只需要检测残差

$$s^k = A_1^{\mathrm{T}} A_2(x_2^{k-1} - x_2^k)$$

是否充分小，这一检测同样也是比较容易的. 综上，当 x_2 更新取到精确解且 $\tau = 1$ 时，判断 ADMM 是否收敛只需要检测前述两个残差 r^k, s^k 是否充分小：

$$\begin{aligned} 0 \approx \|r^k\| &= \|A_1 x_1^k + A_2 x_2^k - b\| \quad \text{（原始可行性）}, \\ 0 \approx \|s^k\| &= \|A_1^{\mathrm{T}} A_2(x_2^{k-1} - x_2^k)\| \quad \text{（对偶可行性）}. \end{aligned} \tag{7.6.11}$$

7.6.2 Douglas-Rachford Splitting 算法

Douglas-Rachford Splitting (DRS) 算法是一类非常重要的算子分裂算法. 它可以用于求解下面的无约束优化问题:

$$\min_x \quad \psi(x) = f(x) + h(x), \tag{7.6.12}$$

其中 f 和 h 是闭凸函数. DRS 算法的迭代格式是

$$x^{k+1} = \mathrm{prox}_{th}(z^k), \tag{7.6.13}$$

$$y^{k+1} = \mathrm{prox}_{tf}(2x^{k+1} - z^k), \tag{7.6.14}$$

$$z^{k+1} = z^k + y^{k+1} - x^{k+1}, \tag{7.6.15}$$

其中 t 是一个正的常数. 我们还可以通过一系列变形来得到 DRS 格式的等价迭代. 首先在原始 DRS 格式中按照 y, z, x 的顺序进行更新，则有

$$\begin{aligned} y^{k+1} &= \mathrm{prox}_{tf}(2x^k - z^k), \\ z^{k+1} &= z^k + y^{k+1} - x^k, \\ x^{k+1} &= \mathrm{prox}_{th}(z^{k+1}). \end{aligned}$$

引入辅助变量 $w^k = z^k - x^k$，并注意到上面迭代中变量 z^k,z^{k+1} 可以消去，则得到 DRS 算法的等价迭代格式

$$y^{k+1} = \mathrm{prox}_{tf}(x^k - w^k), \tag{7.6.16}$$

$$x^{k+1} = \mathrm{prox}_{th}(w^k + y^{k+1}), \tag{7.6.17}$$

$$w^{k+1} = w^k + y^{k+1} - x^{k+1}. \tag{7.6.18}$$

DRS 格式还可以写成关于 z^k 的不动点迭代的形式

$$z^{k+1} = T(z^k), \tag{7.6.19}$$

其中

$$T(z) = z + \mathrm{prox}_{tf}(2\mathrm{prox}_{th}(z) - z) - \mathrm{prox}_{th}(z).$$

将 DRS 格式写成不动点迭代的形式是有好处的：第一，它去掉了迭代中的 x^k,y^k 变量，使得算法形式更加简洁；第二，对不动点迭代的收敛性研究有一些常用的工具和技术手段，例如泛函分析中的压缩映射原理；第三，针对不动点迭代可写出很多种不同类型的加速算法.

下面的定理给出 T 的不动点与 $f + h$ 的极小值之间的关系:

定理 7.8 *(1) 若 z 是(7.6.19) 中 T 的一个不动点，即 $z = T(z)$，则 $x = \mathrm{prox}_{th}(z)$ 是问题 (7.6.12)的一个最小值点.*

(2) 若 x 是问题(7.6.12)的一个最小值点，则存在 $u \in t\partial f(x) \cap (-t\partial h(x))$，且 $x - u = T(x - u)$，即 $x - u$ 是 T 的一个不动点.

证明.

(1) 如果 z 是 T 的一个不动点，即

$$z = T(z) = z + \mathrm{prox}_{tf}(2\mathrm{prox}_{th}(z) - z) - \mathrm{prox}_{th}(z),$$

令 $x = \mathrm{prox}_{th}(z)$，则

$$\mathrm{prox}_{tf}(2x - z) = x = \mathrm{prox}_{th}(z).$$

由邻近算子的定义和最优性条件得

$$x - z \in t\partial f(x), \quad z - x \in t\partial h(x).$$

因此，

$$0 \in t\partial f(x) + t\partial h(x).$$

根据凸优化问题的一阶充要条件知 $x = \mathrm{prox}_{th}(z)$ 是问题(7.6.12)的一个最小值点.

(2) 因为 x 是问题(7.6.12)一个最小值点，根据一阶充要条件，

$$0 \in t\partial f(x) + t\partial h(x),$$

这等价于存在 $u \in t\partial f(x) \cap (-t\partial h(x))$. 由定理 7.2，

$$u \in t\partial f(x) \Longleftrightarrow x = \mathrm{prox}_{tf}(x + u),$$
$$u \in (-t\partial h(x)) \Longleftrightarrow x = \mathrm{prox}_{th}(x - u),$$

然后可以得到

$$\mathrm{prox}_{tf}(2\mathrm{prox}_{th}(x - u) - (x - u)) - \mathrm{prox}_{th}(x - u) = 0,$$

即

$$x - u = T(x - u).$$

□

对于不动点迭代，我们可以通过添加松弛项来加快收敛速度，即

$$z^{k+1} = z^k + \rho(T(z^k) - z^k),$$

其中，当 $1 < \rho < 2$ 时是超松弛，$0 < \rho < 1$ 是欠松弛. 从而得到 DRS 算法的松弛版本

$$\begin{aligned} x^{k+1} &= \mathrm{prox}_{th}(z^k), \\ y^{k+1} &= \mathrm{prox}_{tf}(2x^{k+1} - z^k), \\ z^{k+1} &= z^k + \rho(y^{k+1} - x^{k+1}), \end{aligned}$$

其等价形式为

$$\begin{aligned} y^{k+1} &= \mathrm{prox}_{tf}(x^k - w^k), \\ x^{k+1} &= \mathrm{prox}_{th}((1-\rho)x^k + \rho y^{k+1} + w^k), \\ w^{k+1} &= w^k + \rho y^{k+1} + (1-\rho)x^k - x^{k+1}. \end{aligned}$$

DRS 算法和 ADMM 有一定的等价关系. 考虑本章开始引入的可分的凸问题(7.6.1)：

$$\begin{aligned} \min_{x_1, x_2} \quad & f_1(x_1) + f_2(x_2), \\ \text{s.t.} \quad & A_1 x_1 + A_2 x_2 = b, \end{aligned}$$

它的对偶问题为无约束复合优化问题

$$\min_{z} \quad \underbrace{b^{\mathrm{T}} z + f_1^*(-A_1{}^{\mathrm{T}} z)}_{f(z)} + \underbrace{f_2^*(-A_2{}^{\mathrm{T}} z)}_{h(z)}, \tag{7.6.20}$$

根据问题(7.6.20)的结构拆分出 $f(z)$ 和 $h(z)$，我们对该问题使用 DRS 算法求解. 下面这个定理表明，对原始问题(7.6.1)使用 ADMM 求解就等价于将 DRS 算法应用在对偶问题(7.6.20)上.

定理 7.9 *如果 $w^1 = -tA_2 x_2^0$，那么对问题(7.6.20)应用迭代格式(7.6.16)—(7.6.18)等价于运用 ADMM 到问题(7.6.1).*

证明. 对于迭代格式(7.6.16)，其最优性条件为

$$0 \in tb - tA_1 \partial f_1^*(-A_1^{\mathrm{T}} y^{k+1}) - x^k + w^k + y^{k+1},$$

上式等价于存在 $x_1^k \in \partial f_1^*(-A_1^{\mathrm{T}} y^{k+1})$，使得

$$y^{k+1} = x^k - w^k + t(A_1 x_1^k - b). \tag{7.6.21}$$

根据命题 7.1，$-A_1^{\mathrm{T}} y^{k+1} \in \partial f_1(x_1^k)$，故

$$-A_1^{\mathrm{T}}(x^k - w^k + t(A_1 x_1^k - b)) \in \partial f_1(x_1^k).$$

这就是如下更新

$$x_1^k = \arg\min_{x_1} \left\{ f_1(x_1) + (x^k)^{\mathrm{T}}(A_1 x_1 - b) + \frac{t}{2}\|A_1 x_1 - b - \frac{w^k}{t}\|_2^2 \right\}.$$

的最优性条件.

类似地，迭代(7.6.17)的最优性条件为

$$0 \in tA_2 \partial f_2^*(-A_2^{\mathrm{T}} x^{k+1}) + w^k + y^{k+1} - x^{k+1},$$

其等价于存在 $x_2^k \in \partial f_2^*(-A_2^{\mathrm{T}} x^{k+1})$，使得

$$x^{k+1} = x^k + t(A_1 x_1^k + A_2 x_2^k - b). \tag{7.6.22}$$

同样地，根据命题 7.1，$-A_2^{\mathrm{T}} x^{k+1} \in \partial f_2(x_2^k)$，所以可得

$$-A_2^{\mathrm{T}}(x^k + t(A_1 x_1^k + A_2 x_2^k - b)) \in \partial f_2(x_2^k),$$

其等价于

$$x_2^k = \arg\min_{x_2} \left\{ f_2(x_2) + (x^k)^{\mathrm{T}}(A_2 x_2) + \frac{t}{2}\|A_1 x_1^k + A_2 x_2 - b\|_2^2 \right\}.$$

由(7.6.21)式和(7.6.22)式可得 w-更新转化为 $w^{k+1} = -tA_2 x_2^k$. 令 $z^k = x^{k+1}$，总结上面的更新，可得

$$\begin{aligned}
x_1^k &= \arg\min_{x_1} \left\{ f_1(x_1) + (z^{k-1})^{\mathrm{T}} A_1 x_1 + \frac{t}{2}\|A_1 x_1 + A_2 x_2^{k-1} - b\|^2 \right\}, \\
x_2^k &= \arg\min_{x_2} \left\{ f_2(x_2) + (z^{k-1})^{\mathrm{T}} A_2 x_2 + \frac{t}{2}\|A_1 x_1^k + A_2 x_2 - b\|^2 \right\}, \\
z^k &= z^{k-1} + t\left(A_1 x_1^k + A_2 x_2^k - b\right),
\end{aligned}$$

这就是交替方向乘子法应用到问题 (7.6.1)，其中罚因子 $\rho = t$，步长 $\tau = 1$.

以上论证过程均可以反推，因此等价性成立. □

7.6.3 常见变形和技巧

本小节将给出交替方向乘子法的一些变形以及实现交替方向乘子法的一些技巧.

1. 线性化

我们构造 ADMM 的初衷是将自变量拆分，最终使得关于 x_1 和 x_2 的子问题有显式解. 但是在实际应用中，有时子问题并不容易求解，或者没有必要精确求解. 那么如何寻找子问题的一个近似呢?

不失一般性，我们考虑第一个子问题，即

$$\min_{x_1} \quad f_1(x_1) + \frac{\rho}{2}\|A_1x_1 - v^k\|^2, \tag{7.6.23}$$

其中

$$v^k = b - A_2x_2^k - \frac{1}{\rho}y^k. \tag{7.6.24}$$

当子问题不能显式求解时, 可采用**线性化**的方法 [100] 近似求解问题 (7.6.23). 线性化技巧实际上是使用近似点项对子问题目标函数进行二次近似. 当子问题目标函数可微时，线性化将问题(7.6.23)变为

$$x_1^{k+1} = \arg\min_{x_1}\left\{(\nabla f_1(x_1^k) + \rho A_1^{\mathrm{T}}\left(A_1x_1^k - v^k\right))^{\mathrm{T}}x_1 + \frac{1}{2\eta_k}\|x_1 - x^k\|_2^2\right\},$$

其中 η_k 是步长参数，这等价于做一步梯度下降. 当目标函数不可微时，可以考虑只将二次项线性化，即

$$x_1^{k+1} = \arg\min_{x_1}\left\{f_1(x_1) + \rho\left(A_1^{\mathrm{T}}(A_1x_1^k - v^k)\right)^{\mathrm{T}}x_1 + \frac{1}{2\eta_k}\|x_1 - x^k\|_2^2\right\},$$

这等价于求解子问题(7.6.23)时做一步近似点梯度步. 当然，若 $f_1(x_1)$ 是可微函数与不可微函数的和时，也可将其可微部分线性化.

2. 缓存分解

如果目标函数中含二次函数，例如 $f_1(x_1) = \frac{1}{2}\|Cx_1 - d\|_2^2$，那么针对 x_1 的更新(7.6.5)等价于求解线性方程组

$$(C^{\mathrm{T}}C + \rho A_1^{\mathrm{T}}A_1)x_1 = C^{\mathrm{T}}d + \rho A_1^{\mathrm{T}}v^k,$$

其中 v^k 的定义如(7.6.24)式. 虽然子问题有显式解，但是每步求解的复杂度仍然比较高，这时候可以考虑用**缓存分解**的方法. 首先对 $C^{\mathrm{T}}C+\rho A_1^{\mathrm{T}}A_1$ 进行 Cholesky 分解并缓存分解的结果，在每步迭代中只需要求解简单的三角形方程组；当 ρ 发生更新时，就要重新进行分解. 特别地，当 $C^{\mathrm{T}}C+\rho A_1^{\mathrm{T}}A_1$ 一部分容易求逆，另一部分是低秩的情形时，可以用 SMW 公式来求逆.

3. 优化转移

有时候为了方便求解子问题，可以用一个性质好的矩阵 D 近似二次项 $A_1^{\mathrm{T}}A_1$, 此时子问题(7.6.23)替换为

$$x_1^{k+1}=\arg\min_{x_1}\left\{f_1(x_1)+\frac{\rho}{2}\|A_1x_1-v^k\|_2^2+\frac{\rho}{2}(x_1-x^k)^{\mathrm{T}}(D-A_1^{\mathrm{T}}A_1)(x_1-x^k)\right\},$$

其中 v^k 的定义如(7.6.24)式,这种方法也称为**优化转移**.通过选取合适的 D,当计算 $\arg\min_{x_1}\left\{f_1(x_1)+\frac{\rho}{2}x_1^{\mathrm{T}}Dx_1\right\}$ 明显比计算 $\arg\min_{x_1}\{f_1(x_1)+\frac{\rho}{2}x_1^{\mathrm{T}}A_1^{\mathrm{T}}A_1x_1\}$ 要容易时，优化转移可以极大地简化子问题的计算. 特别地，当 $D=\frac{\eta_k}{\rho}I$ 时，优化转移等价于做单步的近似点梯度步.

4. 二次罚项系数的动态调节

动态调节二次罚项系数在交替方向乘子法的实际应用中是一个非常重要的数值技巧. 在介绍 ADMM 时我们引入了原始可行性和对偶可行性（分别用 $\|r^k\|$ 和 $\|s^k\|$ 度量），见(7.6.11)式. 在实际求解过程中，二次罚项系数 ρ 太大会导致原始可行性 $\|r^k\|$ 下降很快，但是对偶可行性 $\|s^k\|$ 下降很慢；二次罚项系数太小，则会有相反的效果. 这样都会导致收敛比较慢或得到的解的可行性很差. 一个自然的想法是在每次迭代时动态调节惩罚系数 ρ 的大小，从而使得原始可行性和对偶可行性能够以比较一致的速度下降到零. 这种做法通常可以改善算法在实际中的收敛效果，以及使算法表现更少地依赖于惩罚系数的初始选择. 一个简单有效的方式是令

$$\rho^{k+1}=\begin{cases}\gamma_p\rho^k, & \|r^k\|>\mu\|s^k\|,\\ \dfrac{\rho^k}{\gamma_d} & \|s^k\|>\mu\|r^k\|,\\ \rho^k, & \text{其他,}\end{cases}$$

其中 $\mu>1,\gamma_p>1,\gamma_d>1$ 是参数. 常见的选择为 $\mu=10,\gamma_p=\gamma_d=2$. 该惩罚参数更新方式背后的想法是在迭代过程中，将原始可行性 $\|r^k\|$ 和对偶可行

性 $\|s^k\|$ 保持在彼此的 μ 倍内. 如果发现 $\|r^k\|$ 或 $\|s^k\|$ 下降过慢就应该相应增大或减小二次罚项系数 ρ^k. 但在改变 ρ^k 的时候需要注意，若之前利用了缓存分解的技巧，此时分解需要重新计算. 更一般地，我们可以考虑对每一个约束给一个不同的惩罚系数，甚至可以将增广拉格朗日函数(7.6.2)中的二次项 $\frac{\rho}{2}\|r\|^2$ 替换为 $\frac{\rho}{2}r^{\mathrm{T}}Pr$，其中 P 是一个对称正定矩阵. 如果 P 在整个迭代过程中是不变的，我们可以将这个一般的交替方向乘子法解释为将标准的交替方向乘子法应用在修改后的初始问题上——等式约束 $A_1x_1+A_2x_2-b=0$ 替换为 $F(A_1x_1+A_2x_2-b)=0$，其中 F 为 P 的 Cholesky 因子，即 $P=F^{\mathrm{T}}F$，且 F 是对角元为正数的上三角矩阵.

5. 超松弛

另外一种想法是用**超松弛**的技巧，在(7.6.6)式与(7.6.7)式中，$A_1x_1^{k+1}$ 可以被替换为

$$\alpha_k A_1x_1^{k+1}-(1-\alpha_k)(A_2x_2^k-b),$$

其中 $\alpha_k\in(0,2)$ 是一个松弛参数. 当 $\alpha_k>1$ 时，这种技巧称为超松弛；当 $\alpha_k<1$ 时，这种技巧称为欠松弛. 实验表明 $\alpha_k\in[1.5,1.8]$ 的超松弛可以提高收敛速度.

6. 多块与非凸问题的 ADMM

在引入问题(7.6.1)时，我们提到了有两块变量 x_1,x_2. 这个问题不难推广到有多块变量的情形：

$$\begin{aligned}\min_{x_1,x_2,\cdots,x_N}\quad & f_1(x_1)+f_2(x_2)+\cdots+f_N(x_N),\\ \text{s.t.}\quad & A_1x_1+A_2x_2+\cdots+A_Nx_N=b.\end{aligned}\tag{7.6.25}$$

这里 $f_i(x_i)$ 是闭凸函数，$x_i \in \mathbb{R}^{n_i}, A_i \in \mathbb{R}^{m\times n_i}$. 同样可以写出问题(7.6.25)的增广拉格朗日函数 $L_\rho(x_1,x_2,\cdots,x_N,y)$，相应的多块 ADMM 迭代格式为

$$\begin{aligned}
x_1^{k+1} &= \arg\min_x L_\rho(x,x_2^k,\cdots,x_N^k,y^k),\\
x_2^{k+1} &= \arg\min_x L_\rho(x_1^{k+1},x,\cdots,x_N^k,y^k),\\
&\cdots\cdots\cdots\\
x_N^{k+1} &= \arg\min_x L_\rho(x_1^{k+1},x_2^{k+1},\cdots,x,y^k),\\
y^{k+1} &= y^k+\tau\rho(A_1x_1^{k+1}+A_2x_2^{k+1}+\cdots+A_Nx_N^{k+1}-b),
\end{aligned}$$

其中 $\tau \in \left(0,\frac{1}{2}(\sqrt{5}+1)\right)$ 为步长参数.

针对非凸问题，ADMM 格式可能不是良定义的，即每个子问题可能不存在最小值或最小值点不唯一. 若只考虑子问题解存在的情形，我们依然可以形式上利用 ADMM 格式(7.6.5)-(7.6.7)对非凸问题进行求解. 这里 arg min 应该理解为选取子问题最小值点中的一个.

和有两块变量的凸问题上的 ADMM 格式相比，多块（非凸）ADMM 可能不具有收敛性. 但在找到有效算法之前，这两种 ADMM 算法的变形都值得一试. 它们在某些实际问题上也有不错的效果.

7.6.4 应用举例

本小节给出一些交替方向乘子法的应用实例. 在实际中，大多数问题并不直接具有问题(7.6.1) 的形式. 我们需要通过一系列拆分技巧将问题化成 ADMM 的标准形式，同时要求每一个子问题尽量容易求解. 需要指出的是，对同一个问题可能有多种拆分方式，不同方式导出的最终算法可能差异巨大，读者应当选择最容易求解的拆分方式.

1. LASSO 问题

LASSO 问题为

$$\min \quad \mu\|x\|_1+\frac{1}{2}\|Ax-b\|^2.$$

这是典型的无约束复合优化问题，我们可以很容易地将其写成 ADMM 标准问题形式：

$$\begin{aligned} \min_{x,z} \quad & f(x)+h(z), \\ \text{s.t.} \quad & x=z, \end{aligned}$$

其中 $f(x)=\frac{1}{2}\|Ax-b\|^2$，$h(z)=\mu\|z\|_1$. 对于此问题，交替方向乘子法迭代格式为

$$\begin{aligned} x^{k+1} &= \arg\min_x \left\{ \frac{1}{2}\|Ax-b\|^2+\frac{\rho}{2}\|x-z^k+\frac{1}{\rho}y^k\|_2^2 \right\}, \\ &= (A^{\mathrm{T}}A+\rho I)^{-1}(A^{\mathrm{T}}b+\rho z^k-y^k), \\ z^{k+1} &= \arg\min_z \left\{ \mu\|z\|_1+\frac{\rho}{2}\|x^{k+1}-z+\frac{1}{\rho}y^k\|^2 \right\}, \\ &= \mathrm{prox}_{(\mu/\rho)\|\cdot\|_1}\left(x^{k+1}+\frac{1}{\rho}y^k\right), \\ y^{k+1} &= y^k+\tau\rho(x^{k+1}-z^{k+1}). \end{aligned}$$

注意，因为 $\rho>0$，所以 $A^{\mathrm{T}}A+\rho I$ 总是可逆的. x 迭代本质上是计算一个岭回归问题（ℓ_2 范数平方正则化的最小二乘问题）；而对 z 的更新为 ℓ_1 范数的邻近算子，同样有显式解. 在求解 x 迭代时，若使用固定的罚因子 ρ，我们可以缓存矩阵 $A^{\mathrm{T}}A+\rho I$ 的初始分解，从而减小后续迭代中的计算量. 需要注意的是，在 LASSO 问题中，矩阵 $A\in\mathbb{R}^{m\times n}$ 通常有较多的列（即 $m\ll n$），因此 $A^{\mathrm{T}}A\in\mathbb{R}^{n\times n}$ 是一个低秩矩阵，二次罚项的作用就是将 $A^{\mathrm{T}}A$ 增加了一个正定项. 该 ADMM 主要运算量来自更新 x 变量时求解线性方程组，复杂度为 $\mathcal{O}(n^3)$（若使用缓存分解技术或 SMW 公式 (5.5.13) 则可进一步降低每次迭代的运算量）.

接下来考虑 LASSO 问题的对偶问题

$$\begin{aligned} \min \quad & b^{\mathrm{T}}y+\frac{1}{2}\|y\|^2, \\ \text{s.t.} \quad & \|A^{\mathrm{T}}y\|_\infty\leqslant\mu. \end{aligned} \tag{7.6.26}$$

将 $\|A^{\mathrm{T}}y\|_\infty\leqslant\mu$ 变成示性函数放在目标函数中，并引入约束 $A^{\mathrm{T}}y+z=0$，可以得到如下等价问题：

$$\begin{aligned} \min \quad & \underbrace{b^{\mathrm{T}}y+\frac{1}{2}\|y\|^2}_{f(y)}+\underbrace{I_{\|z\|_\infty\leqslant\mu}(z)}_{h(z)}, \\ \text{s.t.} \quad & A^{\mathrm{T}}y+z=0. \end{aligned} \tag{7.6.27}$$

对约束 $A^{\mathrm{T}}y+z=0$ 引入乘子 x，对偶问题的增广拉格朗日函数为

$$L_\rho(y,z,x)=b^{\mathrm{T}}y+\frac{1}{2}\|y\|^2+I_{\|z\|_\infty\leqslant\mu}(z)-x^{\mathrm{T}}(A^{\mathrm{T}}y+z)+\frac{\rho}{2}\|A^{\mathrm{T}}y+z\|^2.$$

在这里我们故意引入符号 x 作为拉格朗日乘子，实际上可以证明（见习题7.11）x 恰好对应的是原始问题的自变量. 以下说明如何求解每个子问题. 当固定 y,x 时，对 z 的更新即向无穷范数球 $\{z|\|z\|_\infty\leqslant\mu\}$ 做欧几里得投影，即将每个分量截断在区间 $[-\mu,\mu]$ 中；当固定 z,x 时，对 y 的更新即求解线性方程组

$$(I+\rho AA^{\mathrm{T}})y=A(x^k-\rho z^{k+1})-b.$$

因此得到 ADMM 迭代格式为

$$\begin{aligned}
z^{k+1}&=\mathcal{P}_{\|z\|_\infty\leqslant\mu}\left(\frac{x^k}{\rho}-A^{\mathrm{T}}y^k\right),\\
y^{k+1}&=(I+\rho AA^{\mathrm{T}})^{-1}\Big(A(x^k-\rho z^{k+1})-b\Big),\\
x^{k+1}&=x^k-\tau\rho(A^{\mathrm{T}}y^{k+1}+z^{k+1}).
\end{aligned}$$

注意，虽然 ADMM 应用于对偶问题也需要求解一个线性方程组，但由于 LASSO 问题的特殊性（$m\ll n$），求解 y 更新的线性方程组需要的计算量是 $\mathcal{O}(m^3)$，使用缓存分解技巧后可进一步降低至 $\mathcal{O}(m^2)$，这大大小于针对原始问题的 ADMM .

对原始问题，另一种可能的拆分方法是

$$\begin{aligned}
\min\quad & \underbrace{\mu\|x\|_1}_{f(x)}+\underbrace{\frac{1}{2}\|z\|^2}_{h(z)},\\
\text{s.t.}\quad & z=Ax-b,
\end{aligned}$$

可以写出其增广拉格朗日函数为

$$L_\rho(x,z,y)=\mu\|x\|_1+\frac{1}{2}\|z\|^2+y^{\mathrm{T}}(Ax-b-z)+\frac{\rho}{2}\|Ax-b-z\|^2.$$

但是如果对这种拆分方式使用 ADMM，求解 x 的更新本质上还是在求解一个 LASSO 问题！这种变形方式将问题绕回了起点，因此并不是一个实用的方法.

我们用同第5.2节中一样的 A 和 b，并取 $\mu=10^{-3}$，分别使用 ADMM 求解原始问题和对偶问题，这里取 $\tau=1.618$，原始问题和对偶问题的参数 ρ 分别为 0.01 和 100，终止条件设为 $|f(x^k)-f(x^{k-1})|<10^{-8}$ 和最大迭代步

数 2000. 此外，对于原始问题的 ADMM，我们还添加终止条件 $\|x^k - z^k\| < 10^{-10}$；相应地，对于对偶问题，额外的终止条件取为 $\|A^{\mathrm{T}}y^k + z^k\| < 10^{-10}$. 算法结果见图7.9. 这里的 ADMM 没有使用连续化策略来调整 μ，因此可以看出 ADMM 相对其他算法的强大之处. 此外，对于这个例子，求解原始问题需要的迭代步数较少，但求解对偶问题每一次迭代所需要的时间更短，综合来看 ADMM 求解对偶问题时更快.

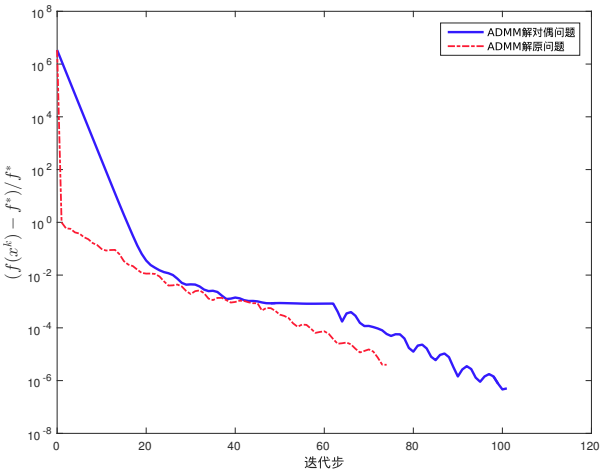


图 7.9 ADMM 求解 LASSO 问题

2. 矩阵分离问题

考虑矩阵分离问题：

$$\begin{aligned} \min_{X,S} \quad & \|X\|_* + \mu\|S\|_1, \\ \text{s.t.} \quad & X + S = M, \end{aligned} \tag{7.6.28}$$

其中 $\|\cdot\|_1$ 与 $\|\cdot\|_*$ 分别表示矩阵 ℓ_1 范数与核范数. 引入乘子 Y 作用在约束 $X + S = M$ 上，我们可以得到此问题的增广拉格朗日函数

$$L_\rho(X,S,Y) = \|X\|_* + \mu\|S\|_1 + \langle Y, X + S - M\rangle + \frac{\rho}{2}\|X + S - M\|_F^2. \tag{7.6.29}$$

在第 $(k+1)$ 步，交替方向乘子法分别求解关于 X 和 S 的子问题来更新得到 X^{k+1} 和 S^{k+1}. 对于 X 子问题，

$$\begin{aligned} X^{k+1} &= \arg\min_X \ L_\rho(X, S^k, Y^k) \\ &= \arg\min_X \left\{ \|X\|_* + \frac{\rho}{2} \left\| X + S^k - M + \frac{Y^k}{\rho} \right\|_F^2 \right\}, \\ &= \arg\min_X \left\{ \frac{1}{\rho}\|X\|_* + \frac{1}{2} \left\| X + S^k - M + \frac{Y^k}{\rho} \right\|_F^2 \right\}, \\ &= U\mathrm{Diag}\Big(\mathrm{prox}_{(1/\rho)\|\cdot\|_1}(\sigma(A)) \Big) V^{\mathrm{T}}, \end{aligned}$$

其中 $A = M - S^k - \dfrac{Y^k}{\rho}$，$\sigma(A)$ 为 A 的所有非零奇异值构成的向量并且 $U\mathrm{Diag}(\sigma(A))V^{\mathrm{T}}$ 为 A 的约化奇异值分解. 对于 S 子问题，

$$\begin{aligned} S^{k+1} &= \arg\min_S \ L_\rho(X^{k+1}, S, Y^k) \\ &= \arg\min_S \left\{ \mu\|S\|_1 + \frac{\rho}{2} \left\| X^{k+1} + S - M + \frac{Y^k}{\rho} \right\|_F^2 \right\} \\ &= \mathrm{prox}_{(\mu/\rho)\|\cdot\|_1} \left(M - X^{k+1} - \frac{Y^k}{\rho} \right). \end{aligned}$$

对于乘子 Y，依然使用常规更新，即

$$Y^{k+1} = Y^k + \tau\rho(X^{k+1} + S^{k+1} - M).$$

那么，交替方向乘子法的迭代格式为

$$\begin{aligned} X^{k+1} &= U\mathrm{Diag}\Big(\mathrm{prox}_{(1/\rho)\|\cdot\|_1}(\sigma(A)) \Big) V^{\mathrm{T}}, \\ S^{k+1} &= \mathrm{prox}_{(\mu/\rho)\|\cdot\|_1} \left(M - L^{k+1} - \frac{Y^k}{\rho} \right), \\ Y^{k+1} &= Y^k + \tau\rho(X^{k+1} + S^{k+1} - M). \end{aligned}$$

3. 全局一致性优化问题

第7.6.1小节介绍了全局一致性优化问题

$$\min_x \quad \sum_{i=1}^N \phi_i(x)$$

并给出了一个拆分方式

$$\begin{aligned} \min_{x_i,z} \quad & \sum_{i=1}^{N}\phi_i(x_i), \\ \text{s.t.} \quad & x_i - z = 0,\ i = 1,2,\cdots,N, \end{aligned}$$

其增广拉格朗日函数为

$$L_\rho(x_1,x_2,\cdots,x_N,z,y_1,y_2,\cdots,y_N) = \sum_{i=1}^{N}\phi_i(x_i) + \sum_{i=1}^{N}y_i^{\mathrm{T}}(x_i - z) + \frac{\rho}{2}\sum_{i=1}^{N}\|x_i - z\|^2.$$

固定 z^k, y_i^k，更新 x_i 的公式为

$$x_i^{k+1} = \arg\min_x \left\{ \phi_i(x) + \frac{\rho}{2}\left\| x - z^k + \frac{y_i^k}{\rho}\right\|^2 \right\}. \tag{7.6.30}$$

在这里注意，虽然表面上看增广拉格朗日函数有 $(N+1)$ 个变量块，但本质上还是两个变量块. 这是因为在更新某 x_i 时并没有利用其他 x_i 的信息，所有 x_i 可以看成一个整体. 相应地，所有乘子 y_i 也可以看成一个整体. 迭代式(7.6.30)的具体计算依赖于 ϕ_i 的形式，在一般情况下更新 x_i 的表达式为

$$x_i^{k+1} = \mathrm{prox}_{\phi_i/\rho}\left(z^k - \frac{y_i^k}{\rho}\right).$$

固定 x_i^{k+1}, y_i^k，问题关于 z 是二次函数，因此可以直接写出显式解：

$$z^{k+1} = \frac{1}{N}\sum_{i=1}^{N}\left(x_i^{k+1} + \frac{y_i^k}{\rho}\right).$$

综上，该问题的交替方向乘子法迭代格式为

$$\begin{aligned} x_i^{k+1} &= \mathrm{prox}_{\phi_i/\rho}\left(z^k - \frac{y_i^k}{\rho}\right),\ i = 1,2,\cdots,N, \\ z^{k+1} &= \frac{1}{N}\sum_{i=1}^{N}\left(x_i^{k+1} + \frac{y_i^k}{\rho}\right), \\ y_i^{k+1} &= y_i^k + \tau\rho(x_i^{k+1} - z^{k+1}),\ i = 1,2,\cdots,N. \end{aligned}$$

4. 非凸集合上的优化问题

非凸集合上的优化问题可以表示为

$$\begin{aligned} \min \quad & f(x), \\ \text{s.t.} \quad & x \in S, \end{aligned} \tag{7.6.31}$$

其中 f 是闭凸函数，但 S 是非凸集合. 利用例 7.15 的技巧，对集合 S 引入示性函数 $I_S(z)$ 并做拆分，可得到问题(7.6.31)的等价优化问题：

$$\begin{aligned} \min \quad & f(x)+I_S(z), \\ \text{s.t.} \quad & x-z=0, \end{aligned} \tag{7.6.32}$$

其增广拉格朗日函数为

$$L_\rho(x,z,y)=f(x)+I_S(z)+y^{\mathrm{T}}(x-z)+\frac{\rho}{2}\|x-z\|^2.$$

由于 f 是闭凸函数，固定 z,y 后对 x 求极小就是计算邻近算子：

$$x^{k+1}=\operatorname{prox}_{f/\rho}\left(z^k-\frac{y^k}{\rho}\right).$$

固定 x,y，对 z 求极小实际上是到非凸集合上的投影问题：

$$z^{k+1}=\arg\min_{z\in S}\frac{1}{2}\left\|z-\left(x^{k+1}+\frac{y^k}{\rho}\right)\right\|^2=\mathcal{P}_S\left(x^{k+1}+\frac{y^k}{\rho}\right).$$

一般来说，由于 S 是非凸集合，计算 $\mathcal{P}_S$ 是比较困难的（例如不能保证存在性和唯一性），但是当 S 有特定结构时，到 S 上的投影可以精确求解.

(1) 基数：如果 $S=\{x\mid \|x\|_0\leqslant c\}$，其中 $\|\cdot\|_0$ 表示 ℓ_0 范数，即非零元素的数目，那么计算任意向量 v 到 S 中的投影就是保留 v 分量中绝对值从大到小排列的前 c 个，其余分量变成 0. 假设 v 的各个分量满足

$$|v_{i_1}|\geqslant|v_{i_2}|\geqslant\cdots\geqslant|v_{i_n}|,$$

则投影算子可写成

$$(\mathcal{P}_S(v))_i=\begin{cases} v_i, & i\in\{i_1,i_2,\cdots,i_c\}, \\ 0, & \text{其他}. \end{cases}$$

(2) 低秩投影：当变量 $x\in\mathbb{R}^{m\times n}$ 是矩阵时，一种常见的非凸约束是在低秩矩阵空间中进行优化问题求解，即 $S=\{x\mid \operatorname{rank}(x)\leqslant r\}$. 此时到低秩矩阵空间上的投影等价于对 x 做截断奇异值分解. 设 x 的奇异值分解为

$$x=\sum_{i=1}^{\min\{m,n\}}\sigma_i u_i v_i^{\mathrm{T}},$$

其中 $\sigma_1 \geqslant \cdots \geqslant \sigma_{\min\{m,n\}} \geqslant 0$ 为奇异值，u_i, v_i 为对应的左右奇异向量，则 $\mathcal{P}_S$ 的表达式为

$$\mathcal{P}_S(x) = \sum_{i=1}^{r} \sigma_i u_i v_i^{\mathrm{T}}.$$

(3) 布尔(Bool)约束：如果限制变量 x 只能在 $\{0,1\}$ 中取值，即 $S = \{0,1\}^n$，容易验证 $\mathcal{P}_S(v)$ 就是简单地把 v 的每个分量 v_i 变为 0 和 1 中离它更近的数，一种可能的实现方式是分别对它们做四舍五入操作.

5. 多块交替方向乘子法的反例

这里给出一个多块交替方向乘子法的例子，并且从数值上说明若直接采用格式(7.6.25)，则算法未必收敛.

考虑最优化问题

$$\begin{aligned} \min \quad & 0, \\ \text{s.t.} \quad & A_1x_1 + A_2x_2 + A_3x_3 = 0, \end{aligned} \tag{7.6.33}$$

其中 $A_i \in \mathbb{R}^3$, $i = 1,2,3$ 为三维空间中的非零向量，$x_i \in \mathbb{R}$, $i = 1,2,3$ 是自变量. 问题(7.6.33)实际上就是求解三维空间中的线性方程组，若 A_1, A_2, A_3 之间线性无关，则问题(7.6.33) 只有零解. 此时容易计算出最优解对应的乘子为 $y = (0,0,0)^{\mathrm{T}}$.

现在推导多块交替方向乘子法的格式. 问题(7.6.33)的增广拉格朗日函数为

$$L_\rho(x,y) = 0 + y^{\mathrm{T}}(A_1x_1 + A_2x_2 + A_3x_3) + \frac{\rho}{2}\|A_1x_1 + A_2x_2 + A_3x_3\|^2.$$

当固定 x_2, x_3, y 时，对 x_1 求最小可推出

$$A_1^{\mathrm{T}}y + \rho A_1^{\mathrm{T}}(A_1x_1 + A_2x_2 + A_3x_3) = 0,$$

整理可得

$$x_1 = -\frac{1}{\|A_1\|^2}\left(A_1^{\mathrm{T}}\left(\frac{y}{\rho} + A_2x_2 + A_3x_3\right)\right).$$

可类似地计算 x_2, x_3 的表达式，因此多块交替方向乘子法的迭代格式可以写

为

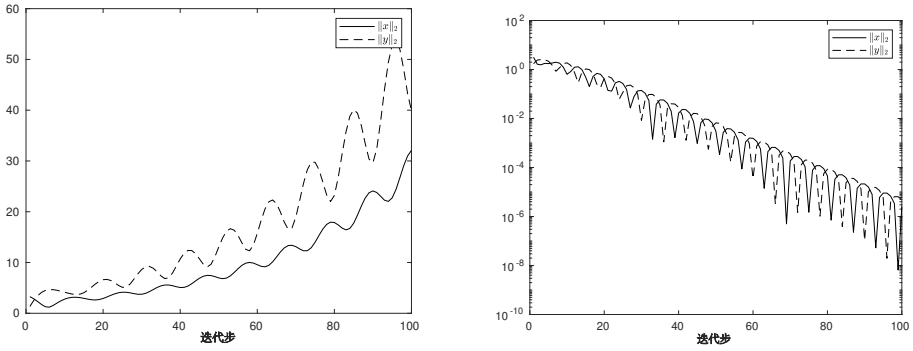
$$\begin{aligned} x_1^{k+1} &= -\frac{1}{\|A_1\|^2} A_1^{\mathrm{T}} \left(\frac{y^k}{\rho} + A_2 x_2^k + A_3 x_3^k \right), \\ x_2^{k+1} &= -\frac{1}{\|A_2\|^2} A_2^{\mathrm{T}} \left(\frac{y^k}{\rho} + A_1 x_1^{k+1} + A_3 x_3^k \right), \\ x_3^{k+1} &= -\frac{1}{\|A_3\|^2} A_3^{\mathrm{T}} \left(\frac{y^k}{\rho} + A_1 x_1^{k+1} + A_2 x_2^{k+1} \right), \\ y^{k+1} &= y^k + \rho(A_1 x_1^{k+1} + A_2 x_2^{k+1} + A_3 x_3^{k+1}). \end{aligned} \tag{7.6.34}$$

对此问题而言，罚因子取不同值仅仅是将乘子 y^k 缩放了常数倍，所以罚因子 ρ 的任意取法（包括动态调节）都是等价的．在数值实验中我们不妨取 $\rho = 1$.

格式(7.6.34)的收敛性与 A_i, $i = 1,2,3$ 的选取有关．为了方便，令 $A = [A_1, A_2, A_3]$，以及 $x = (x_1, x_2, x_3)^{\mathrm{T}}$，并选取 A 为

$$\widetilde{A} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 2 \end{bmatrix} \quad \text{或} \quad \widehat{A} = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

在迭代中，自变量初值初值选为 $(1,1,1)$，乘子选为 $(0,0,0)$．图 7.10 记录了在两种不同 A 条件下，x 和 y 的 ℓ_2 范数随迭代的变化过程．可以看到，当



(a) 系数矩阵为 $\widetilde{A}$　　(b) 系数矩阵为 $\widehat{A}$

图 7.10　选取不同 A 时的数值结果

$A = \widetilde{A}$ 时数值结果表明迭代是发散的，而当 $A = \widehat{A}$ 时数值结果表明迭代是收敛的．另一个比较有趣的观察是，在图7.10（b）中 $\|x\|$ 与 $\|y\|$ 并不是单

调下降的，而是在下降的同时有规律地振荡. 实际上，文献 [30] 中具体解释了 A 取 $\widetilde{A}$ 会导致发散的原因.

7.7 随机优化算法

随着大数据时代的来临，以及机器学习和深度学习等人工智能领域的发展，许多大规模的优化问题随之产生，它们对传统优化理论和算法都产生了巨大的挑战. 幸运的是，这些问题往往与概率和统计学科有很大联系，由此促使了随机优化算法的广泛使用. 随机算法的思想可以追溯到 Monro-Robbins 算法 [110]，相比传统优化算法，随机优化算法能极大地节省每步迭代的运算量，从而使得算法在大规模数据中变得可行. 本节将介绍随机梯度算法的基本形式和收敛性理论，以及当前在深度学习领域广泛应用的一些随机梯度型算法.

为了方便理解本节主要考虑的问题形式，首先介绍机器学习的一个基本模型——监督学习模型. 假定 (a,b) 服从概率分布 P，其中 a 为输入，b 为标签. 我们的任务是要给定输入 a 预测标签 b，即要决定一个最优的函数 ϕ 使得期望风险 $\mathbb{E}[L(\phi(a),b)]$ 最小，其中 $L(\cdot,\cdot)$ 表示损失函数，用来衡量预测的准确度，函数 ϕ 为某个函数空间中的预测函数. 在实际问题中我们并不知道真实的概率分布 P，而是随机采样得到的一个数据集 $\mathcal{D} = \{(a_1,b_1),(a_2,b_2),\cdots,(a_N,b_N)\}$. 然后我们用经验风险来近似期望风险，并将预测函数 $\phi(\cdot)$ 参数化为 $\phi(\cdot;x)$ 以缩小要找的预测函数的范围，即要求解下面的极小化问题：

$$\min_{x} \quad \frac{1}{N}\sum_{i=1}^{N} L(\phi(a_i;x),b_i). \tag{7.7.1}$$

对应于随机优化问题 (3.4.1)，有 $\xi_i = (a_i,b_i)$ 以及

$$f_i(x) = L(\phi(a_i;x),b_i), \quad h(x) = 0.$$

在机器学习中，大量的问题都可以表示为问题 (7.7.1) 的形式. 由于数据规模巨大，计算目标函数的梯度变得非常困难，但是可以通过采样的方式只计算部分样本的梯度来进行梯度下降，往往也能达到非常好的数值表现，而每步的运算量却得到了极大的减小.

我们将主要考虑如下随机优化问题：

$$\min_{x\in\mathbb{R}^n} \quad f(x) \overset{\text{def}}{=\!=} \frac{1}{N}\sum_{i=1}^{N} f_i(x), \tag{7.7.2}$$

其中 $f_i(x)$ 对应第 i 个样本的损失函数. 问题 (7.7.2) 也称为随机优化问题的有限和形式.

7.7.1 随机梯度下降算法

下面为了讨论方便，先假设 (7.7.2) 中每一个 $f_i(x)$ 是凸的、可微的. 因此可以运用梯度下降算法

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k), \tag{7.7.3}$$

来求解原始的优化问题. 在迭代格式(7.7.3)中，

$$\nabla f(x^k) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x^k).$$

在绝大多数情况下，我们不能通过化简的方式得到 $\nabla f(x^k)$ 的表达式，要计算这个梯度必须计算出所有的 $\nabla f_i(x^k), i = 1,2,\cdots,N$ 然后将它们相加. 然而在机器学习中，采集到的样本量是巨大的，因此计算 $\nabla f(x^k)$ 需要非常大的计算量. 使用传统的梯度法求解机器学习问题并不是一个很好的做法.

既然梯度的计算很复杂，有没有减少计算量的方法呢? 这就是下面要介绍的随机梯度下降算法 (SGD). 它的基本迭代格式为

$$x^{k+1} = x^k - \alpha_k \nabla f_{s_k}(x^k), \tag{7.7.4}$$

其中 s_k 是从 $\{1,2,\cdots,N\}$ 中随机等可能地抽取的一个样本，α_k 称为步长[1]. 通过对比 (7.7.3) 式和 (7.7.4) 式可知，随机梯度算法不去计算全梯度 $\nabla f(x^k)$，而是从众多样本中随机抽出一个样本 s_i，然后仅仅计算这个样本处的梯度 $\nabla f_{s_k}(x^k)$，以此作为 $\nabla f(x^k)$ 的近似. 注意，在全梯度 $\nabla f(x^k)$ 的表达式中含系数 $\frac{1}{N}$，而迭代格式(7.7.4)中不含 $\frac{1}{N}$. 这是因为我们要保证随机梯度的条件期望恰好是全梯度，即

$$\mathbb{E}_{s_k}[\nabla f_{s_k}(x^k)|x^k] = \nabla f(x^k).$$

这里使用条件期望的符号 $\mathbb{E}[\cdot|x^k]$ 的原因是迭代点 x^k 本身也是一个随机变量. 实际计算中每次只抽取一个样本 s_k 的做法比较极端，常用的形式是小批量（mini-batch）随机梯度法，即随机选择一个元素个数很少的集合

[1]在机器学习和深度学习领域中，更多的时候被称为学习率 (learning rate)

$\mathcal{I}_k \subset \{1,2,\cdots,N\}$，然后执行迭代格式

$$x^{k+1} = x^k - \frac{\alpha_k}{|\mathcal{I}_k|}\sum_{s\in\mathcal{I}_k}\nabla f_s(x^k),$$

其中 $|\mathcal{I}_k|$ 表示 $\mathcal{I}_k$ 中的元素个数. 本节后面的阐述中虽然只考虑最简单形式的随机梯度下降算法 (7.7.4)，但很多变形和分析都可以推广到小批量随机梯度法.

随机梯度下降法使用一个样本点的梯度代替了全梯度，并且每次迭代选取的样本点是随机的，这使得每次迭代时计算梯度的复杂度变为了原先的 $\frac{1}{N}$，在样本量 N 很大的时候无疑是一个巨大的改进. 但正因为如此，算法中也引入了随机性，一个自然的问题是这样的算法还会有收敛性吗？如果收敛，是什么意义下的收敛？这些问题将在第 7.7.3 小节中给出具体的回答.

当 $f_i(x)$ 是凸函数但不一定可微时，我们可以用 $f_i(x)$ 的次梯度代替梯度进行迭代. 这就是随机次梯度算法，它的迭代格式为

$$x^{k+1} = x^k - \alpha_k g^k, \tag{7.7.5}$$

其中 α_k 为步长，$g^k \in \partial f_{s_k}(x^k)$ 为随机次梯度，其期望为真实的次梯度.

随机梯度算法在深度学习中得到了广泛的应用，下面介绍其在深度学习中的一些变形.

1. 动量方法

传统的梯度法在问题比较病态时收敛速度非常慢，随机梯度下降法也有类似的问题. 为了克服这一缺陷，人们提出了动量方法（momentum），旨在加速学习. 该方法在处理高曲率或是带噪声的梯度上非常有效，其思想是在算法迭代时一定程度上保留之前更新的方向，同时利用当前计算的梯度调整最终的更新方向. 这样一来，可以在一定程度上增加稳定性，从而学习得更快，并且还有一定摆脱局部最优解的能力. 从形式上来看，动量方法引入了一个速度变量 v，它代表参数移动的方向和大小. 动量方法的具体迭代格式如下：

$$v^{k+1} = \mu_k v^k - \alpha_k \nabla f_{s_k}(x^k), \tag{7.7.6}$$

$$x^{k+1} = x^k + v^{k+1}. \tag{7.7.7}$$

在计算当前点的随机梯度 $\nabla f_{s_i}(x^k)$ 后，我们并不是直接将其更新到变量 x^k 上，即不完全相信这个全新的更新方向，而是将其和上一步更新方向 v^k 做

线性组合来得到新的更新方向 v^{k+1}. 由动量方法迭代格式立即得出当 $\mu_k = 0$ 时该方法退化成随机梯度下降法. 在动量方法中，参数 μ_k 的范围是 $[0,1)$，通常取 $\mu_k \geqslant 0.5$，其含义为迭代点带有较大惯性，每次迭代会在原始迭代方向的基础上做一个小的修正. 在普通的梯度法中，每一步迭代只用到了当前点的梯度估计，动量方法的更新方向还使用了之前的梯度信息. 当许多连续的梯度指向相同的方向时，步长就会很大，这从直观上看也是非常合理的.

图 7.11 比较了梯度法和动量方法在例 5.2 中的表现. 可以看到普通梯度法生成的点列会在椭圆的短轴方向上来回移动，而动量方法生成的点列更快收敛到了最小值点.

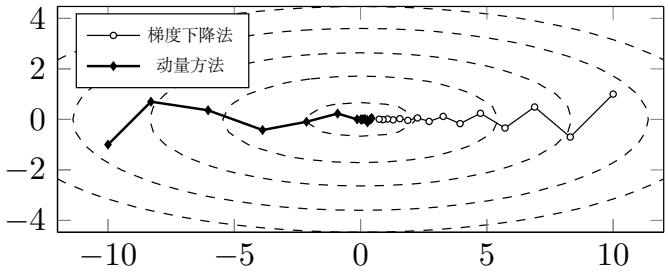


图 7.11 动量方法在海瑟矩阵病态条件下的表现

2. Nesterov 加速算法

针对光滑问题的 Nesterov 加速算法迭代的随机版本为

$$y^{k+1} = x^k + \mu_k(x^k - x^{k-1}), \tag{7.7.8}$$

$$x^{k+1} = y^{k+1} - \alpha_k \nabla f_{s_k}(y^{k+1}), \tag{7.7.9}$$

其中 $\mu_k = \dfrac{k-1}{k+2}$，步长 α_k 是一个固定值或者由线搜索确定. 现在我们通过一些等价变形来说明 Nesterov 加速算法可以看成是某种动量方法. 首先在第 k 步迭代引入速度变量

$$v^k = x^k - x^{k-1},$$

再合并原始 Nesterov 加速算法的两步迭代可以得到

$$x^{k+1} = x^k + \mu_k(x^k - x^{k-1}) - \alpha_k \nabla f_k(x^k + \mu_k(x^k - x^{k-1})).$$

如果定义有关 v^{k+1} 的迭代式

$$v^{k+1} = \mu_k v^k - \alpha_k \nabla f_k(x^k + \mu_k v^k),$$

则得到关于 x^k 和 v^k 的等价迭代：

$$v^{k+1} = \mu_k v^k - \alpha_k \nabla f_{s_k}(x^k + \mu_k v^k), \tag{7.7.10}$$

$$x^{k+1} = x^k + v^{k+1}. \tag{7.7.11}$$

与动量方法相比，二者的主要差别在梯度的计算上. Nesterov 加速算法先对点施加速度的作用，再求梯度，这可以理解为对标准动量方法做了一个校正.

3. AdaGrad

在一般的随机梯度法中，调参是一个很大的难点，参数设置的好坏对算法的性能有显著的影响. 所以我们希望算法能在运行的过程中，根据当前情况自发地调整参数，这就是 AdaGrad(adaptive subgradient methods) 的出发点. 对无约束光滑凸优化问题，点 x 是问题的解等价于该点处梯度为零向量. 但对大部分问题而言，梯度的每个分量收敛到零的速度是不同的. 传统梯度算法只有一个统一的步长 α_k 来调节每一步迭代，它没有针对每一个分量考虑. 当梯度的某个分量较大时，可以推断出在该方向上函数变化比较剧烈，此时应该用小步长；当梯度的某个分量较小时，在该方向上函数比较平缓，此时应该用大步长. AdaGrad 就是根据这个思想设计的.

令 $g^k = \nabla f_{s_k}(x^k)$，为了记录整个迭代过程中梯度各个分量的累积情况，引入向量

$$G^k = \sum_{i=1}^{k} g^i \odot g^i.$$

从 G^k 的定义可知 G^k 的每个分量表示在迭代过程中，梯度在该分量处的累积平方和. 当 G^k 的某分量较大时，我们认为该分量变化比较剧烈，因此应采用小步长，反之亦然. 因此 AdaGrad 的迭代格式为

$$x^{k+1} = x^k - \frac{\alpha}{\sqrt{G^k + \varepsilon \mathbf{1}_n}} \odot g^k, \tag{7.7.12}$$

$$G^{k+1} = G^k + g^{k+1} \odot g^{k+1}, \tag{7.7.13}$$

这里 $\frac{\alpha}{\sqrt{G^k + \varepsilon \mathbf{1}_n}}$ 中的除法和求根运算都是对向量每个分量分别操作的（下同），α 为初始步长，引入 $\varepsilon \mathbf{1}_n$ 这一项是为了防止除零运算. 可以看到 AdaGrad

的步长大致反比于历史梯度累计值的算术平方根，所以梯度较大时步长下降很快，反之则下降较慢，这样做的效果就是在参数空间更平缓的方向上，前后两次迭代的距离较大. 在凸优化问题中 AdaGrad 有比较好的理论性质，但实际应用中也发现在训练深度神经网络模型时，从训练开始就积累梯度平方会导致步长过早或过多减小.

如果在 AdaGrad 中使用真实梯度 $\nabla f(x^k)$，那么 AdaGrad 也可以看成是一种介于一阶和二阶的优化算法. 考虑 $f(x)$ 在点 x^k 处的二阶泰勒展开：

$$f(x) \approx f(x^k) + \nabla f(x^k)^{\mathrm{T}}(x - x^k) + \frac{1}{2}(x - x^k)^{\mathrm{T}} B^k (x - x^k),$$

我们知道选取不同的 B^k 可以导出不同的优化算法. 例如使用常数倍单位矩阵近似 B^k 时可得到梯度法；利用海瑟矩阵作为 B^k 时可得到牛顿法. 而 AdaGrad 则是使用一个对角矩阵来作为 B^k. 具体地，取

$$B^k = \frac{1}{\alpha}\mathrm{Diag}(\sqrt{G^k + \varepsilon \mathbf{1}_n})$$

时导出的算法就是 AdaGrad. 读者可自行验证.

4. RMSProp

RMSProp（root mean square propagation）是对 AdaGrad 的一个改进，该方法在非凸问题上可能表现更好. AdaGrad 会累加之前所有的梯度分量平方，这就导致步长是单调递减的，因此在训练后期步长会非常小，同时这样做也加大了计算的开销. 所以 RMSProp 提出只需使用离当前迭代点比较近的项，同时引入衰减参数 ρ. 具体地，令

$$M^{k+1} = \rho M^k + (1 - \rho) g^{k+1} \odot g^{k+1},$$

再对其每个分量分别求根，就得到均方根 (root mean square)

$$R^k = \sqrt{M^k + \varepsilon \mathbf{1}_n}, \tag{7.7.14}$$

最后将均方根的倒数作为每个分量步长的修正，得到 RMSProp 迭代格式：

$$x^{k+1} = x^k - \frac{\alpha}{R^k} \odot g^k, \tag{7.7.15}$$

$$M^{k+1} = \rho M^k + (1 - \rho) g^{k+1} \odot g^{k+1}. \tag{7.7.16}$$

引入参数 ε 同样是为了防止分母为 0 的情况发生. 一般取 $\rho = 0.9, \alpha = 0.001$. 可以看到 RMSProp 和 AdaGrad 的唯一区别是将 G^k 替换成了 M^k.

5. AdaDelta

AdaDelta 在 RMSProp 的基础上，对历史的 Δx^k 也同样累积平方并求均方根：

$$D^k = \rho D^{k-1} + (1-\rho)\Delta x^k \odot \Delta x^k, \tag{7.7.17}$$

$$T^k = \sqrt{D^k + \varepsilon \mathbf{1}_n}, \tag{7.7.18}$$

然后使用 T^{k-1} 和 R^k 的商对梯度进行校正，完整的过程如算法 7.11 所示，其中 T^k 和 R^k 的定义分别为 (7.7.18) 式和 (7.7.14)式.

算法 7.11 AdaDelta

1. 输入 x^1, ρ, ε.
2. 置初值 $M^0 = 0, D^0 = 0$.
3. **for** $k = 1,2,\cdots,K$ **do**
4. 随机选取 $i \in \{1,2,\cdots,N\}$，计算梯度 $g^k = \nabla f_i(x^k)$.
5. 计算 $M^k = \rho M^{k-1} + (1-\rho) g^k \odot g^k$.
6. 计算 $\Delta x^k = -\dfrac{T^{k-1}}{R^k} \odot g^k$.
7. 计算 $D^k = \rho D^{k-1} + (1-\rho)\Delta x^k \odot \Delta x^k$.
8. $x^{k+1} \leftarrow x^k + \Delta x^k$.
9. **end for**

注意，计算步长时 T 和 R 的下标相差 1，这是因为我们还没有计算出 Δx^k 的值，无法使用 T^k 进行计算. AdaDelta 的特点是步长选择较为保守，同时也改善了 AdaGrad 步长单调下降的缺陷.

6. Adam

Adam（adaptive moment estimation）本质上是带动量项的 RMSProp，它利用梯度的一阶矩估计和二阶矩估计动态调整每个参数步长.虽然 RMSProp 也采用了二阶矩估计，但是缺少修正因子，所以它在训练初期可能有比较大的偏差. Adam 的优点主要在于经过偏差修正后，每一次迭代的步长都有一个确定范围，使得参数比较平稳.

与 RMSProp 和动量方法相比，Adam 可以看成是带修正的二者的结合. 在 Adam 中不直接使用随机梯度作为基础的更新方向，而是选择了一个动

量项进行更新：

$$S^k = \rho_1 S^{k-1} + (1-\rho_1) g^k.$$

于此同时它和 RMSProp 类似，也会记录迭代过程中梯度的二阶矩：

$$M^k = \rho_2 M^{k-1} + (1-\rho_2) g^k \odot g^k.$$

与原始动量方法和 RMSProp 的区别是，由于 S^k 和 M^k 本身带有偏差，Adam 不直接使用这两个量更新，而是在更新前先对其进行修正：

$$\hat{S}^k = \frac{S^k}{1-\rho_1^k}, \quad \hat{M}^k = \frac{M^k}{1-\rho_2^k},$$

这里 ρ_1^k, ρ_2^k 分别表示 ρ_1, ρ_2 的 k 次方．Adam 最终使用修正后的一阶矩和二阶矩进行迭代点的更新．

$$x^{k+1} = x^k - \frac{\alpha}{\sqrt{\hat{M}^k + \varepsilon \mathbf{1}_n}} \odot \hat{S}^k.$$

我们将完整的迭代过程整理到算法 7.12 中．这里参数 ρ_1 通常选为 0.9，

算法 7.12 Adam

1. 给定步长 α，矩估计的指数衰减速率 ρ_1, ρ_2，x^1.
2. 置初值 $S^0 = 0$，$M^0 = 0$.
3. **for** $k = 1, 2, \cdots, K$ **do**
4. 随机选取 $i \in \{1, 2, \cdots, N\}$，计算梯度 $g^k = \nabla f_i(x^k)$.
5. 更新一阶矩估计：$S^k = \rho_1 S^{k-1} + (1-\rho_1) g^k$.
6. 更新二阶矩估计：$M^k = \rho_2 M^{k-1} + (1-\rho_2) g^k \odot g^k$.
7. 修正一阶矩的偏差：$\hat{S}^k = \dfrac{S^k}{1-\rho_1^k}$.
8. 修正二阶矩的偏差：$\hat{M}^k = \dfrac{M^k}{1-\rho_2^k}$.
9. $x^{k+1} = x^k - \dfrac{\alpha}{\sqrt{\hat{M}^k + \varepsilon \mathbf{1}_n}} \odot \hat{S}^k$.
10. **end for**

ρ_2 选为 0.999，全局步长 $\alpha = 0.001$.

上面的很多算法已经被实现在主流的深度学习框架中，可以非常方便地用于训练神经网络：Pytorch 里实现的算法有 AdaDelta，AdaGrad，Adam，Nesterov，RMSProp 等；Tensorflow 里实现的算法则有 AdaDelta，AdaGradDA，AdaGrad，ProximalAdagrad，Ftrl，Momentum，Adam 和 CenteredRMSProp 等.

7.7.2 应用举例

随机优化在机器学习的监督模型中有非常多的应用，本小节介绍两个例子——逻辑回归和神经网络.

1. 逻辑回归

逻辑回归是最基本的线性分类模型，它经常用来作为各种分类模型的比较标准. 给定数据集 $\{(a_i,b_i)\}_{i=1}^N$，带 ℓ_2 范数平方正则项的逻辑回归对应的优化问题 (7.7.2) 可以写成如下形式：

$$\min_{x\in\mathbb{R}^n} \quad f(x)=\frac{1}{N}\sum_{i=1}^N f_i(x)=\frac{1}{N}\sum_{i=1}^N \ln(1+\exp(-b_i\cdot a_i^{\mathrm{T}}x))+\lambda\|x\|_2^2,$$

其中

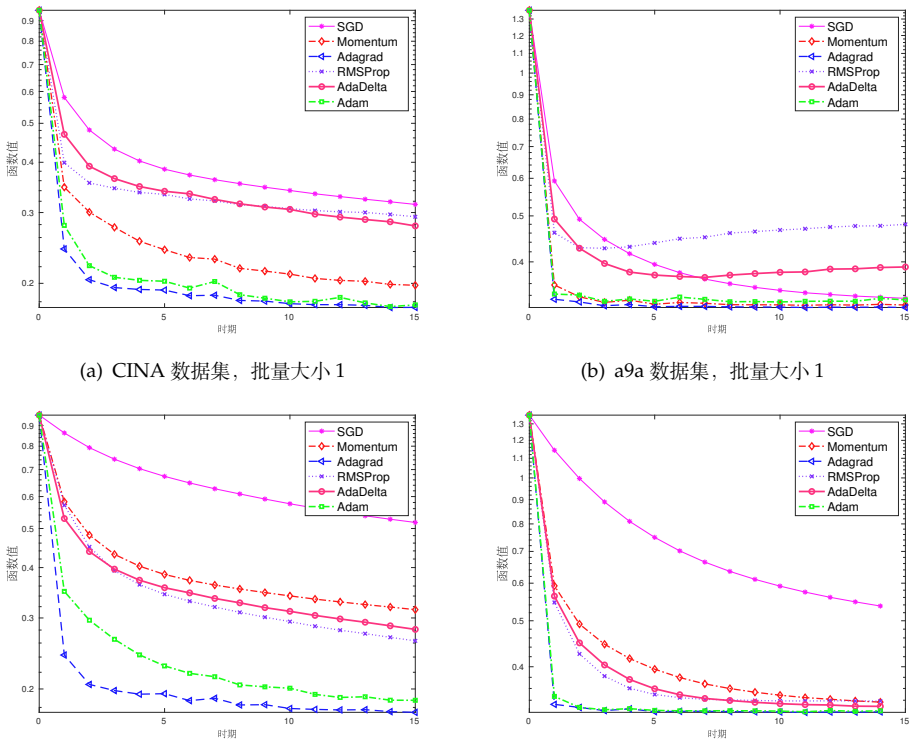
$$f_i(x)=\ln(1+\exp(-b_i\cdot a_i^{\mathrm{T}}x))+\lambda\|x\|_2^2.$$

每步我们随机取一个下标 i_k 对应的梯度 $\nabla f_{i_k}(x^k)$ 做随机梯度下降，其迭代格式可以写成

$$x^{k+1}=x^k-\alpha_k\nabla f_{i_k}(x^k)=x^k-\alpha_k\left(\frac{-\exp(-b_{i_k}\cdot a_{i_k}^{\mathrm{T}}x^k)b_{i_k}a_{i_k}}{1+\exp(-b_{i_k}\cdot a_{i_k}^{\mathrm{T}}x^k)}+2\lambda x^k\right),$$

其中 i_k 为从 $\{1,2,\cdots,N\}$ 中随机抽取的一个样本，α_k 为步长.

这里和第 5.4 节一样，采用 LIBSVM[29] 网站的数据集并令 $\lambda=\dfrac{10^{-2}}{N}$，分别测试不同随机算法在数据集 CINA 和 a9a 上的表现. 我们采用网格搜索方法来确定随机算法中的参数值，对每个参数重复 5 次数值实验并取其平均表现. 对比发现：对随机梯度算法，步长 $\alpha_k=10^{-3}$ 表现最好. 动量方法中取 $\mu_k=0.8,\alpha_k=10^{-3}$；Adagrad, RMSProp 和 Adam 中的 α 分别取为 $0.4,10^{-3},5\cdot10^{-3}$，数值稳定参数均设置为 $\varepsilon=10^{-7}$. 数值结果见图7.12. 随机算法的批量大小（batch size）表示每次迭代所采用的样本数（即随机生成的下标 i 的个数）. 在横坐标中，每个时期（epoch）表示计算了 N 次分量函数 f_i 的梯度. 可以看到，加入了动量之后，随机梯度法的收敛加快，但没有自适应类梯度方法快. 值得注意的是，当批量大小从 1 变成 10 时，随机梯度法和动量方法达到相同精度需要的时期数变多，但大的批量大小对应的算法效率更高（计算梯度时矩阵乘法的并行效率以及内存利用率会更高）. 在自适应梯度类方法中，AdaGrad 对该凸问题收敛最快，Adam 次之. 在 a9a 数据集上，RMSProp 和 AdaDelta 在批量大小为 1 时尾部出现函数值上



(a) CINA 数据集，批量大小 1

(b) a9a 数据集，批量大小 1

(c) CINA 数据集，批量大小 10

(d) a9a 数据集，批量大小 10

图 7.12 使用不同类型的随机梯度法求解逻辑回归问题.

升（尽管设置更小的 α 可以避免出现上升，但会大大降低目标函数值的下降速度），批量大小为 10 时算法产生的 x 对应的目标函数值更小.

2. 多层感知机神经网络

多层感知机也叫全连接神经网络，是一种基本的网络结构，第 1.3 节已经简要地介绍了这一模型. 考虑有 L 个隐藏层的多层感知机，给定输入 $a \in \mathbb{R}^p$，则多层感知机的输出可用如下迭代过程表示：

$$y^{(l)} = t(x^{(l)} y^{(l-1)} + w^{(l)}), \quad l = 1, 2, \cdots, L+1,$$

其中 $x^{(l)} \in \mathbb{R}^{m_{l-1} \times m_l}$ 为系数矩阵，$w^{(l)} \in \mathbb{R}^{m_l}$ 为非齐次项，$t(\cdot)$ 为非线性激活函数. 该感知机的输出为 $y^{(L+1)}$.

现在用非线性函数 $h(a;x)$ 来表示该多层感知机，其中

$$x = (x^{(1)}, x^{(2)}, \cdots, x^{(L)}, w^{(1)}, w^{(2)}, \cdots, w^{(L)})$$

表示所有网络参数的集合，则学习问题可以表示成经验损失函数求极小问题：

$$\min \quad \frac{1}{N}\sum_{i=1}^{N} L(h(a_i;x), b_i).$$

同样地，由于目标函数表示成了样本平均的形式，我们可以用随机梯度算法：

$$x^{k+1} = x^k - \tau_k \nabla_x L(h(a_{s_k};x^k), b_{s_k}),$$

其中 s_k 为从 $\{1,2,\cdots,N\}$ 中随机抽取的一个样本．算法最核心的部分为求梯度，由于函数具有复合结构，因此可以采用后传算法．假定已经得到关于第 l 隐藏层的导数 $\dfrac{\partial L}{\partial y^{(l)}}$，然后可以通过下面递推公式得到关于第 l 隐藏层参数的导数以及关于前一个隐藏层的导数：

$$\begin{aligned} \frac{\partial L}{\partial w^{(l)}} &= \frac{\partial L}{\partial y^{(l)}} \odot \frac{\partial t}{\partial z}, \\ \frac{\partial L}{\partial x^{(l)}} &= \left(\frac{\partial L}{\partial y^{(l)}} \odot \frac{\partial t}{\partial z}\right)(y^{(l-1)})^{\mathrm{T}}, \\ \frac{\partial L}{\partial y^{(l-1)}} &= (x^{(l)})^{\mathrm{T}}\left(\frac{\partial L}{\partial y^{(l)}} \odot \frac{\partial t}{\partial z}\right). \end{aligned} \tag{7.7.19}$$

其中 $\odot$ 为逐元素相乘．完整的后传算法见算法 7.13.

算法 7.13 后传算法

1. $g \leftarrow \nabla_{\hat{y}} L(\hat{y}, b_{s_k})$.
2. **for** $l = L+1, L, \cdots, 1$. **do**
3. $\quad g \leftarrow g \odot \dfrac{\partial t}{\partial z}$.
4. $\quad \dfrac{\partial L}{\partial w^{(l)}} = g$.
5. $\quad \dfrac{\partial L}{\partial x^{(l)}} = g(y^{(l-1)})^{\mathrm{T}}$.
6. $\quad g \leftarrow (x^{(l)})^{\mathrm{T}} g$.
7. **end for**

7.7.3 收敛性分析

随机梯度算法具有不确定性，这样的算法会有收敛性吗？本小节将简要回答这一问题. 概括来说，随机梯度下降法的收敛性依赖于步长的选取以及函数 f 本身的性质，在不同条件下会有不同结果. 这一点和梯度法是类似的. 为了简单起见我们仅介绍目标函数为可微强凸函数的结果.

首先我们列出这一部分的统一假设.

假设 7.2 对问题(7.7.2)使用迭代算法(7.7.4)时，

(1) $f(x)$ 是可微函数，每个 $f_i(x)$ 梯度存在；

(2) $f(x)$ 是梯度利普希茨连续的，相应常数为 L；

(3) $f(x)$ 是强凸函数，强凸参数为 μ；

(4) 随机梯度二阶矩是一致有界的，即存在 M，对任意的 $x \in \mathbb{R}^n$ 以及随机下标 s^k，有

$$\mathbb{E}_{s_k}[\|\nabla f_{s_k}(x)\|^2] \leqslant M^2 < +\infty.$$

下面的定理表明，如果设置递减的步长，收敛阶可以达到 $\mathcal{O}\left(\frac{1}{K}\right)$.

定理 7.10 (随机梯度算法的收敛速度 [17]定理 4.7) 在假设 7.2 的条件下，取递减的步长

$$\alpha_k = \frac{\beta}{k+\gamma},$$

其中 $\beta > \frac{1}{2\mu}$，$\gamma > 0$，使得 $\alpha_1 \leqslant \frac{1}{2\mu}$，那么对于任意的 $k \geqslant 1$，都有

$$\mathbb{E}[f(x^k) - f(x^*)] \leqslant \frac{L}{2}\mathbb{E}[\Delta_k^2] \leqslant \frac{L}{2}\frac{v}{\gamma + k}, \tag{7.7.20}$$

这里 $v = \max\left\{\frac{\beta^2 M^2}{2\beta\mu - 1}, (\gamma+1)\Delta_1^2\right\}$.

定理 7.10 表明对于强凸函数，随机梯度下降法的收敛速度可以达到 $\mathcal{O}\left(\frac{1}{k}\right)$. 对于一般的凸函数随机梯度算法也有一定的收敛性，为此我们在表 7.1 比较随机算法和普通算法的复杂度. 这里复杂度是指计算梯度的次数，ε 代表希望达到的精度，N 表示样本数量. 对于普通梯度下降方法，每一次迭代都需要计算 N 次梯度，而随机算法只需要计算一次. 我们可以看到次

梯度算法的普通版本和随机版本的收敛速度并没有差别，而每步的计算量能大大降低. 但是在可微光滑和强凸情形下，随机梯度算法的收敛速度要慢于梯度算法. 下一小节将讨论产生这一差别的原因，并介绍方差减小技术来改进它.

表 7.1 梯度下降法的算法复杂度

	f 凸 (次梯度算法)	f 可微强凸	f 可微强凸且 L-光滑
随机算法	$\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$\mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{1}{\varepsilon}\right)$
普通算法	$\mathcal{O}\left(\frac{N}{\varepsilon^2}\right)$	$\mathcal{O}\left(\frac{N}{\varepsilon}\right)$	$\mathcal{O}\left(N\ln\left(\frac{1}{\varepsilon}\right)\right)$

7.7.4 方差减小技术

上一小节证明了在次梯度情形以及 $f(x)$ 可微强凸的条件下，随机梯度方法的算法复杂度要小于普通的梯度法. 但是我们也发现随机方法容易受到梯度估计噪声的影响，这使得它在固定步长下不能收敛，并且在递减步长下也只能达到 R-次线性收敛 $\mathcal{O}\left(\frac{1}{k}\right)$，而普通梯度法在强凸且梯度 L-利普希茨连续的条件下可以达到 Q-线性收敛速度. 下面分析这两种算法的主要区别.

在假设 7.2 的条件下，对梯度下降法有

$$\begin{aligned}\Delta_{k+1}^2 &= \|x^{k+1}-x^*\|^2 = \|x^k-\alpha\nabla f(x^k)-x^*\|^2\\ &= \Delta_k^2 - 2\alpha\langle\nabla f(x^k), x^k-x^*\rangle + \alpha^2\|\nabla f(x^k)\|^2\\ &\leqslant (1-2\alpha\mu)\Delta_k^2 + \alpha^2\|\nabla f(x^k)\|_2^2 \qquad (\mu\text{-强凸})\\ &\leqslant (1-2\alpha\mu+\alpha^2L^2)\Delta_k^2. \qquad (L\text{-光滑})\end{aligned} \tag{7.7.21}$$

对随机梯度下降法，利用条件期望的性质（具体细节读者可自行验证）有

$$\begin{aligned}\mathbb{E}[\Delta_{k+1}^2] &= \mathbb{E}[\|x^{k+1}-x^*\|_2^2] = \mathbb{E}[\|x^k-\alpha\nabla f_{s_k}(x^k)-x^*\|]^2\\ &= \mathbb{E}[\Delta_k^2] - 2\alpha\mathbb{E}[\langle\nabla f_{s_k}(x^k), x^k-x^*\rangle] + \alpha^2\mathbb{E}[\|\nabla f_{s_k}(x^k)\|^2]\\ &= \mathbb{E}[\Delta_k^2] - 2\alpha\mathbb{E}[\langle\nabla f(x^k), x^k-x^*\rangle] + \alpha^2\mathbb{E}[\|\nabla f_{s_k}(x^k)\|^2]\\ &\leqslant (1-2\alpha\mu)\mathbb{E}[\Delta_k^2] + \alpha^2\mathbb{E}[\|\nabla f_{s_k}(x^k)\|^2] \qquad (\mu\text{-强凸})\\ &= (1-2\alpha\mu)\mathbb{E}[\Delta_k^2] + \alpha^2\mathbb{E}[\|\nabla f_{s_k}(x^k)-\nabla f(x^k)+\nabla f(x^k)\|^2]\\ &\leqslant (1-2\alpha\mu+\alpha^2L^2)\mathbb{E}[\Delta_k^2] + \alpha^2\mathbb{E}[\|\nabla f_{s_k}(x^k)-\nabla f(x^k)\|^2],\end{aligned}$$

也即

$$\mathbb{E}[\Delta_{k+1}^2] \leqslant \underbrace{(1-2\alpha\mu+\alpha^2L^2)\mathbb{E}[\Delta_k^2]}_{A} + \underbrace{\alpha^2\mathbb{E}[\|\nabla f_{s_k}(x^k)-\nabla f(x^k)\|^2]}_{B}. \quad (7.7.22)$$

从上面的分析中可以看到两种算法的主要差别就在 B 项上，也就是梯度估计的某种方差．它导致了随机梯度算法只能有 $\mathcal{O}\left(\frac{1}{k}\right)$ 的收敛速度．但是在许多机器学习的应用中，随机梯度算法的真实的收敛速度可能会更快一些．这主要是因为许多应用对解的精度要求并没有太高，而在开始部分方差较小，即有 $B \ll A$，那么我们会观察到近似 Q-线性收敛速度；而随着迭代步数增多，方差逐渐增大，算法最终的收敛速度为 $\mathcal{O}\left(\frac{1}{k}\right)$．所以为了能获得比较快的渐进收敛速度，我们的主要目标就是减少方差项 B 来加快随机梯度算法的收敛速度．下面将要介绍三种减小方差的算法：

- SAG (stochastic average gradient)
- SAGA[2]
- SVRG (stochastic variance reduced gradient)

相对于普通的随机梯度算法使用更多的样本点，这些算法的基本思想都是通过利用之前计算得到的信息来减小方差，最终获得 Q-线性收敛速度．

1. SAG 算法和 SAGA 算法

在随机梯度下降法中，每一步迭代仅仅使用了当前点的随机梯度，而迭代计算的历史随机梯度则直接丢弃不再使用．当迭代接近收敛时，上一步的随机梯度同样也是当前迭代点处梯度的一个很好的估计．随机平均梯度法 (SAG) 就是基于这一想法构造的．在迭代中，SAG 算法记录所有之前计算过的随机梯度，再与当前新计算的随机梯度求平均，最终作为下一步的梯度估计．具体来说，SAG 算法在内存中开辟了存储 N 个随机梯度的空间

$$[g_1^k, g_2^k, \cdots, g_N^k],$$

分别用于记录和第 i 个样本相关的最新的随机梯度．在第 k 步更新时，若抽取的样本点下标为 s_k，则计算随机梯度后将 $g_{s_k}^k$ 的值更新为当前的随机梯度

[2]和 SAG 算法不同，SAGA 算法名字本身并不是四个英文单词的缩写.

值，而其他未抽取到的下标对应的 g_i^k 保持不变. 每次 SAG 算法更新使用的梯度方向是所有 g_i^k 的平均值. 它的数学迭代格式为

$$x^{k+1} = x^k - \frac{\alpha_k}{N}\sum_{i=1}^{N} g_i^k, \tag{7.7.23}$$

其中 g_i^k 的更新方式为

$$g_i^k = \begin{cases} \nabla f_{s_k}(x^k), & i = s_k, \\ g_i^{k-1}, & \text{其他}, \end{cases} \tag{7.7.24}$$

这里 s_k 是第 k 次迭代随机抽取的样本. 由 g_i^k 的更新方式不难发现，每次迭代时只有一个 g_i^k 的内容发生了改变，而其他的 g_i^k 是直接沿用上一步的值. 因此 SAG 迭代公式还可以写成

$$x^{k+1} = x^k - \alpha_k \left(\frac{1}{N}(\nabla f_{s_k}(x^k) - g_{s_k}^{k-1}) + \frac{1}{N}\sum_{i=1}^{N} g_i^{k-1} \right), \tag{7.7.25}$$

其中 g_i^k 的更新方式如(7.7.24)式. 在 SAG 算法中 $\{g_i^k\}$ 的初值可简单地取为零向量或中心化的随机梯度向量，我们不加证明地指出，SAG 算法每次使用的随机梯度的条件期望并不是真实梯度 $\nabla f(x^k)$，但随着迭代进行，随机梯度的期望和真实梯度的偏差会越来越小.

接下来直接给出 SAG 算法的收敛性分析.

定理 7.11 (SAG 算法的收敛性 [114]) *在假设 7.2 的条件下，取固定步长 $\alpha_k = \dfrac{1}{16L}$，$g_i^k$ 的初值取为零向量，则对任意的 k，我们有*

$$\mathbb{E}[f(x^k)] - f(x^*) \leqslant \left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^k C_0,$$

其中常数 C_0 为与 k 无关的常数.

上述定理表明 SAG 算法确实有 Q-线性收敛速度. 但是 SAG 算法的缺点在于需要存储 N 个梯度向量，当样本量 N 很大时，这无疑是一个很大的开销. 因此 SAG 算法在实际中很少使用，它的主要价值在于算法的思想. 很多其他实用算法都是根据 SAG 算法变形而来的.

SAGA 算法是 SAG 算法的一个修正. 我们知道 SAG 算法每一步的随机梯度的条件期望并不是真实梯度，这其实是一个缺陷. SAGA 算法则使用无偏的梯度向量作为更新方向，它的迭代方式为

$$x^{k+1} = x^k - \alpha_k \left(\nabla f_{s_k}(x^k) - g_{s_k}^{k-1} + \frac{1}{N}\sum_{i=1}^{N} g_i^{k-1} \right). \tag{7.7.26}$$

对比(7.7.25)式可以发现，SAGA 算法去掉了 $\nabla f_{s_k}(x^k) - g_{s_k}^{k-1}$ 前面的系数 $\frac{1}{N}$. 可以证明每次迭代使用的梯度方向都是无偏的，即

$$\mathbb{E}\left[\nabla f_{s_k}(x^k) - g_{s_k}^{k-1} + \frac{1}{N}\sum_{i=1}^{N} g_i^{k-1} \,\middle|\, x^k\right] = \nabla f(x^k).$$

SAGA 算法同样有 Q-线性收敛速度，实际上我们有如下结果：

定理 7.12 (SAGA 算法的收敛性 [35]) 在假设 7.2 的条件下，取固定步长 $\alpha_k = \frac{1}{2(\mu N + L)}$. 定义 $\Delta_k = \|x^k - x^*\|$，则对任意的 $k \geqslant 1$ 有

$$\mathbb{E}[\Delta_k^2] \leqslant \left(1 - \frac{\mu}{2(\mu N + L)}\right)^k \left(\Delta_1^2 + \frac{N(f(x^1) - f(x^*))}{\mu N + L}\right). \tag{7.7.27}$$

如果强凸的参数 μ 是未知的，也可以取 $\alpha = \frac{1}{3L}$，此时有类似的收敛结果.

2. SVRG 算法

与 SAG 算法和 SAGA 算法不同，SVRG 算法通过周期性缓存全梯度的方法来减小方差. 具体做法是在随机梯度下降方法中，每经过 m 次迭代就设置一个检查点，计算一次全梯度，在之后的 m 次迭代中，将这个全梯度作为参考点来达到减小方差的目的. 令 $\tilde{x}^j$ 是第 j 个检查点，则我们需要计算点 $\tilde{x}^j$ 处的全梯度

$$\nabla f(\tilde{x}^j) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\tilde{x}^j),$$

在之后的迭代中使用方向 v^k 作为更新方向：

$$v^k = \nabla f_{s_k}(x^k) - (\nabla f_{s_k}(\tilde{x}^j) - \nabla f(\tilde{x}^j)), \tag{7.7.28}$$

其中 $s_k \in \{1,2,\cdots,N\}$ 是随机选取的一个样本. 注意到给定 $s_1, s_2, \cdots, s_{k-1}$ 时 $x^k, \tilde{x}^j$ 均为定值，由 v^k 的表达式可知

$$\begin{aligned}\mathbb{E}[v^k | s_1, s_2, \cdots, s_{k-1}] &= \mathbb{E}[\nabla f_{s_k}(x^k) | x^k] - \mathbb{E}[\nabla f_{s_k}(\tilde{x}^j) - \nabla f(\tilde{x}^j) | s_1, s_2, \cdots, s_{k-1}] \\ &= \nabla f(x^k) - 0 = \nabla f(x^k),\end{aligned}$$

即 v^k 在条件期望的意义下是 $\nabla f(x^k)$ 的一个无偏估计. 公式(7.7.28)有简单的直观理解. 事实上，我们希望用 $\nabla f_{s_k}(\tilde{x}^j)$ 去估计 $\nabla f(\tilde{x}^j)$，那么 $\nabla f_{s_k}(\tilde{x}^j) -$

$\nabla f(\tilde{x}^j)$ 就可以看作梯度估计的误差，所以在每一步随机梯度迭代用该项来对 $\nabla f_{s_k}(x^k)$ 做一个校正.

接下来分析方差，并通过数学的方法说明为什么选取参考点会使得估计的方差变小. 这里需要作一个额外的假设

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leqslant L\|x-y\|, \quad i = 1,2,\cdots,N,$$

也即 $f(x)$ 的每一个子函数都是梯度 L-利普希茨连续的. 为记号方便，令 $y = \tilde{x}^j$，x^* 为 $f(x)$ 的最小值点，$\Delta_k = \|x^k - x^*\|$ 为 x^k 与 x^* 的距离，则

$$\begin{aligned}
&\mathbb{E}\left[\|v^k\|^2\right] = \mathbb{E}\left[\|\nabla f_{s_k}(x^k) - (\nabla f_{s_k}(y) - \nabla f(y))\|^2\right] \\
&= \mathbb{E}\left[\|\nabla f_{s_k}(x^k) - \nabla f_{s_k}(y) + \nabla f(y) + \nabla f_{s_k}(x^*) - \nabla f_{s_k}(x^*)\|^2\right] \\
&\leqslant 2\mathbb{E}\left[\|\nabla f_{s_k}(x^k) - \nabla f_{s_k}(x^*)\|^2\right] + 2\mathbb{E}\left[\|\nabla f_{s_k}(y) - \nabla f(y) - \nabla f_{s_k}(x^*)\|^2\right] \\
&\leqslant 2L^2\mathbb{E}\left[\Delta_k^2\right] + 2\mathbb{E}\left[\|\nabla f_{s_k}(y) - \nabla f_{s_k}(x^*)\|^2\right] \\
&\leqslant 2L^2\mathbb{E}\left[\Delta_k^2\right] + 2L^2\mathbb{E}\left[\|y - x^*\|^2\right].
\end{aligned} \tag{7.7.29}$$

其中第一个不等式是因为 $\|a+b\|^2 \leqslant 2\|a\|^2 + 2\|b\|^2$，第二个不等式使用了有关二阶矩的不等式

$$\mathbb{E}\left[\|\xi - \mathbb{E}\xi\|^2\right] \leqslant \mathbb{E}\left[\|\xi\|^2\right].$$

从上面的不等式可以看出，如果 x^k 和 y 非常接近 x^*，梯度估计的方差就很小. 显然频繁地更新 y 可以使得方差更小，但是这样同时也增加了计算全梯度的次数.

算法 7.14 中给出了完整的 SVRG 算法，在这里注意为了之后推导收敛性的方便，我们将 SVRG 算法写成了二重循环的形式. 在 SVRG 方法中，只给出了 $\tilde{x}^j$ 的一种更新方法，实际上 $\tilde{x}^j$ 可以有其他的选法，例如取随机选取前 m 步的 x^k 中的一个. 与 SAG 算法和 SAGA 算法不同，SVRG 算法不需要分配存储空间来记录 N 个梯度向量，但它的代价在于每 m 步都要计算一次全梯度，在每次迭代的时候也需要多计算一个梯度 $\nabla f_{s_k}(\tilde{x}^j)$.

下面给出 SVRG 算法的收敛性. 这里的收敛性是针对参考点序列 $\{\tilde{x}^j\}$ 而言的.

定理 7.13 (SVRG 算法的收敛性 [76]) *设每个 $f_i(x)$ 是可微凸函数，且梯度为 L-利普希茨连续的，函数 $f(x)$ 是强凸的，强凸参数为 μ. 在算法 7.14*

算法 7.14 SVRG 算法

1. 给定 $\tilde{x}^0$，步长 α，更新次数 m.
2. 计算全梯度 $\nabla f(x^0)$.
3. **for** $j = 1,2,\cdots,J$ **do**
4. 赋值 $y = \tilde{x}^{j-1}$, $x^1 = \tilde{x}^{j-1}$.
5. 计算全梯度 $\nabla f(y)$.
6. **for** $k = 1,2,\cdots,m$ **do**
7. 随机选取 $s_k \in \{1,2,\cdots,N\}$.
8. 计算 $v^k = \nabla f_{s_k}(x^k) - (\nabla f_{s_k}(y) - \nabla f(y))$.
9. 更新 $x^{k+1} = x^k - \alpha v^k$.
10. **end for**
11. 计算参考点 $\tilde{x}^j = \dfrac{1}{m}\sum_{i=1}^{m} x^i$.
12. **end for**

中取步长 $\alpha \in \left(0, \dfrac{1}{2L}\right]$，并且 m 充分大使得

$$\rho = \frac{1}{\mu\alpha(1-2L\alpha)m} + \frac{2L\alpha}{1-2L\alpha} < 1, \tag{7.7.30}$$

那么 *SVRG* 算法对于参考点 $\tilde{x}^j$ 在函数值期望的意义下有 Q-线性收敛速度：

$$\mathbb{E}f(\tilde{x}^j) - f(x^*) \leqslant \rho\mathbb{E}[f(\tilde{x}^{j-1}) - f(x^*)]. \tag{7.7.31}$$

7.8 总结

本章介绍了众多求解复合优化问题的算法，这些算法可以应用到常见的大部分凸优化问题上. 针对非凸问题，适用的算法有近似点梯度法、分块坐标下降法、交替方向乘子法以及随机优化算法. Nesterov 加速算法和近似点算法在经过合适变形后也可推广到一些非凸问题上. 由于在非凸情形下原始问题和对偶问题之间可能缺乏明显的关系，对偶算法应用在此类问题上比较困难. 读者需要在应用算法之前判断优化问题的种类，之后选择合适的算法求解.

近似点梯度法是解决非光滑、无约束、规模较大问题的常用算法. 通常情况下它能利用问题的结构，有着比次梯度法更好的表现. 有关近似点梯

度法更进一步的讨论我们推荐读者阅读文献 [102]. 近似点梯度算法还有一些变形，比如镜像下降算法 [7]，条件梯度算法 [48,74]，惯性近似点梯度法 [99]，有兴趣的读者可以自行了解. 近似点算法可以理解成一种特殊的近似点梯度算法，也可以理解成次梯度算法的隐式格式. 同时，近似点算法与增广拉格朗日函数法有着非常密切的关系，由于其等价性，增广拉格朗日函数法可以视作（次）梯度算法隐式格式的一种实现方式.

Nesterov 加速算法能够对近似点梯度算法进行加速，对于凸复合优化问题目标函数的收敛速度能够从 $\mathcal{O}\left(\frac{1}{k}\right)$ 的速度提高到 $\mathcal{O}\left(\frac{1}{k^2}\right)$. 但 Nesterov 算法最初是如何构造出来的并没有一个很直观的解释，人们只能将这一贡献归功于 Nesterov 本人强大的数学推导能力. 直观理解 Nesterov 加速算法的本质是一个很有意思的课题，例如，文献 [118] 的作者提出使用微分方程的观点来解释 Nesterov 加速算法. 通过理解 Nesterov 加速算法的背后原理，人们或许能够构造出其他非平凡的加速算法. 本章主要给出了常用的一些 Nesterov 加速算法，有关 FISTA 算法的部分参考了 [8-9]，第二类 Nesterov 加速算法在 [96] 中提出，第三类 Nesterov 加速算法可参考 [11,95].

分块坐标下降法的历史可以追溯到 1960 年之前，它由于算法结构简单且易于实现而受到一些应用数学家的关注. 然而当时分块坐标下降法并不是一个很流行的算法，由于其过于简单、收敛速度缓慢，人们的研究重点大多在其他有较快收敛速度的算法上. 分块坐标下降法的复兴主要是由统计学习和机器学习推动的，这些问题的特点是自变量维数高，而且不需要求解太精确，因此分块坐标下降法非常适用于处理大规模问题. 在分块坐标下降算法中，每个变量块更新的顺序对算法的表现有很大影响，常用的做法是循环更新、随机更新以及选取梯度模长最大的分量更新等，随机更新的分块坐标下降法还可衍生出异步并行算法，详见 [10,83-84,109]. 其他的一些有关分块坐标下降法的综述可参考 [128].

当原始问题求解过于困难时，其对偶问题可能比较容易求解. 对偶算法就是利用这种思想设计的算法. 在算法的构造当中，共轭函数起到了非常关键的作用. 有关共轭函数的性质读者可参考 [69] 的第十章，[16] 的第七章以及 [112]. 我们没有介绍原始 – 对偶混合梯度法 (PDHG) 的收敛性，实际上 PDHG 收敛需要更强的条件，更详细的讨论推荐读者阅读 [65-66].

早期的交替方向乘子法可以参考 1975 年左右针对变分问题提出的算法 [49,53]，其在 21 世纪初产生了大量的相关应用. 因为其结构简单、易于实现，交替方向乘子法很快受到人们的青睐. 但该方法本身也具有一定的缺

陷，例如正文给出了多块变量的交替方向乘子法的反例，该反例来自 [30]. 为了解决这一问题，人们提出 RP-ADMM[119]，即如果每一轮迭代的乘子更新前随机独立地改变多块变量的更新顺序，那么这种算法在求解上述问题时在期望意义下收敛. 此后 RP-ADMM 被拓展到不可分的多块凸优化问题上 [31]. 正文还提到交替方向乘子法在一般情况下每一步可能不是良定义的，因此人们提出了近似点交替方向乘子法 [42]，即在更新 x_1 与 x_2 子问题目标函数中再添加正定二次项. 近几十年也有许多文献对交替方向乘子法的收敛性有进一步讨论，读者可参考文献 [37,67,70-71].

随机优化的早期研究可参考 1951 年的 Monro-Robbins 算法 [110]. 最近一些年，随着大数据和深度学习等相关领域的发展，随机优化算法受到了人们的大量关注和研究. 我们这里介绍了无约束光滑优化问题的随机梯度法以及相应的方差减少和加速技术. 除了梯度方法外，人们也研究了随机拟牛顿法 [15,24,93,130] 和带有子采样的牛顿法 [12,22-23,88,104,131]. 对于目标函数中带有非光滑项的情形，也有相应的随机近似点梯度类算法，见 [115,129]. 对于约束优化问题（例如随机主成分分析）的随机梯度法，感兴趣的读者可以参考 [75,138].

本章的部分内容基于 Lieven Vandenberghe 教授的课件编写，包含近似点梯度法、Nesterov 加速算法、近似点算法、对偶分解算法、交替方向乘子法. 分块坐标下降法结构叙述主要参考了 [132]，相关的收敛性分析主要参考了 [14]. Chambolle-Pock 算法的收敛性编写参考了 [28]. 交替方向乘子法方面的内容同时参考了印卧涛教授的课件，相关的收敛性分析主要参考了 [32].

习题 8

7.1 证明例7.2中的运算法则成立.

7.2 求下列函数的邻近算子：

(a) $f(x) = I_C(x)$，其中 $C = \{(x,t) \in \mathbb{R}^{n+1} | \|x\|_2 \leqslant t\}$；

(b) $f(x) = \inf\limits_{y \in C} \|x - y\|$，其中 C 是闭凸集；

(c) $f(x) = \dfrac{1}{2}(\inf\limits_{y \in C} \|x - y\|)^2$，其中 C 是闭凸集.

7.3 对一般复合优化问题的加速算法（算法 7.8），试证明：

(a) 当 $t_k = \gamma_k \lambda_k$ 且 $h(x) = 0$ 时，算法 7.8 等价于第二类 Nesterov 加速算法；

(b) 当 $t_k = \lambda_k$ 时，算法 7.8 等价于近似点梯度法.

7.4 假设 f 是闭凸函数，证明 Moreau 分解的推广成立，即对任意的 $\lambda > 0$ 有

$$x = \operatorname{prox}_{\lambda f}(x) + \lambda \operatorname{prox}_{\lambda^{-1} f^*}\left(\frac{x}{\lambda}\right) \quad \forall x.$$

提示：利用 *Moreau* 分解的结论.

7.5 写出关于 LASSO 问题的鞍点问题形式，并写出原始 – 对偶混合梯度算法和 Chambolle-Pock 算法.

7.6 设函数 $f(x_1, x_2) = |x_1 - x_2| - \min\{x_1, x_2\}$，其定义域为 $[0,1] \times [0,1]$. 试推导基于格式(7.4.2)的分块坐标下降法（x_1 和 x_2 分别看做一个变量块），此算法是否收敛?

7.7 试对分组 LASSO 问题（即例7.5）推导出基于格式(7.4.3)的分块坐标下降法.

7.8 考虑约束优化问题

$$\begin{aligned} \min \quad & \max\{e^{-x} + y, y^2\}, \\ \text{s.t.} \quad & y \geqslant 2, \end{aligned}$$

其中 $x, y \in \mathbb{R}$ 为自变量.

(a) 通过引入松弛变量 z，试说明该问题等价于

$$\begin{aligned} \min \quad & \max\{e^{-x} + y, y^2\} + I_{\mathbb{R}_+}(z), \\ \text{s.t.} \quad & y - z = 2; \end{aligned}$$

(b) 推导 (a) 中问题的对偶问题，并求出原始问题的最优解；

(c) 对 (a) 中的问题形式，使用 ADMM 求解时可能会遇到什么问题?

7.9 写出对于线性规划对偶问题运用 ADMM 的迭代格式，以及与之等价的对于原始问题的 DRS 格式，并指出 ADMM 和 DRS 算法更新变量之间的关系.

7.10 考虑 ℓ_0 范数优化问题的罚函数形式：

$$\min \quad \lambda\|x\|_0 + \frac{1}{2}\|Ax-b\|^2,$$

其中 $A \in \mathbb{R}^{m\times n}, m < n$ 为实矩阵，$\|\cdot\|_0$ 为 ℓ_0 范数，即非零元素的个数. 试针对 ℓ_0 范数优化问题形式化推导具有两个变量块的 ADMM 格式. 在算法中每个子问题是如何求解的?

7.11 试说明 LASSO 对偶问题中，若在问题(7.6.27)中对约束

$$A^T y + z = 0$$

引入乘子 x，则 x 恰好对应 LASSO 原始问题(7.6.26)中的自变量.

7.12 实现关于 LASSO 问题使用以下算法的程序，并比较它们的效率

(a) 近似点梯度算法；

(b) Nesterov 加速算法；

(c) 交替方向乘子法；

(d) Chambolle-Pock 算法；

(e) 分块坐标下降法；

(f) 随机近似点梯度算法.

7.13 设 $f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x)$, 其中每个 $f_i(x)$ 是可微函数, 且 $f(x)$ 为梯度 L-利普希茨连续的. $\{x^k\}$ 是由随机梯度下降法产生的迭代序列，s_k 为第 k 步随机抽取的下标. 证明：

$$\mathbb{E}[\|\nabla f_{s_k}(x^k)\|^2] \leqslant L^2\mathbb{E}[\|x^k - x^*\|^2] + \mathbb{E}[\|\nabla f_{s_k}(x^k) - \nabla f(x^k)\|^2],$$

其中 x^* 是 $f(x)$ 的一个最小值点.

7.14 在 SAGA 算法中，每一步的下降方向取为

$$v^k = \nabla f_{s_k}(x^k) - g_{s_k}^{k-1} + \frac{1}{N}\sum_{i=1}^{N} g_i^{k-1},$$

假设初值 $g_i^0 = 0, i = 1,2,\cdots,N$，证明：

$$\mathbb{E}[v^k | s_1, s_2, \cdots, s_{k-1}] = \nabla f(x^k).$$

附录 A　符号表

符号	含义
$\mathbb{R}$	实数集合
$\bar{\mathbb{R}}$	广义实数集合
$\mathbb{R}^n$	n 维欧几里得空间
$\mathbb{R}^{m\times n}$	$m \times n$ 矩阵空间
$\mathcal{S}^n$	n 阶对称矩阵空间
$\mathcal{S}^n_+$	n 阶对称半正定矩阵空间
$\mathcal{S}^n_{++}$	n 阶对称正定矩阵空间
$\mathbf{int}\, S$	集合 S 的内点
$\mathbf{affine}\, S$	集合 S 的仿射包
$\mathbf{conv}\, S$	集合 S 的凸包
$A \subseteq B$	集合 A 是 B 的子集
$A \subset B$	集合 A 是 B 的真子集
$A + B$	集合 A 与 B 的加法（分别从 A 与 B 中选取一个元素相加构成的集合）
$A - B$	集合 A 与 B 的减法（分别从 A 与 B 中选取一个元素相减构成的集合）
$N_\delta(x)$	点 x 处半径为 δ 的邻域
$\mathrm{dist}(x, S)$	点 x 到集合 S 的欧几里得距离
$a \overset{\text{def}}{=\!=} b$	表达式 a 的定义为表达式 b
x	最优化问题的变量
x^k	最优化算法在第 k 步时自变量 x 的值
x^*	最优化问题的最优解
x^{T}	向量或矩阵的转置

符号	含义
$\bar{x}^{\mathrm{T}}$	向量或矩阵的共轭转置
$\mathbf{dom}\, f$	函数 f 的定义域
$\mathbf{epi}\, f$	函数 f 的上方图
$\nabla f(x)$	函数 f 在点 x 处的梯度
$\nabla^2 f(x)$	函数 f 在点 x 处的海瑟矩阵
$\partial f(x;d)$	函数 f 在点 x 处关于方向 d 的方向导数
$\partial f(x)$	函数 f 在点 x 处的次微分
$\arg\min_{x\in\mathcal{X}} f(x)$	函数 f 在可行域 $\mathcal{X}$ 中的一个最小值点
$\arg\max_{x\in\mathcal{X}} f(x)$	函数 f 在可行域 $\mathcal{X}$ 中的一个最大值点
$\mathrm{sign}(x)$	符号函数
$\ln(x)$	自然对数（以 e 为底）
d^k	最优化算法在第 k 步时的下降方向
α_k	最优化算法在第 k 步时的步长
$x \geqslant 0$	向量 x 每一个分量都大于或等于 0，即 $x_i \geqslant 0, i = 1,2,\cdots,n$
$x \odot y$	向量或矩阵 x 和 y 的 Hadamard 积（逐分量相乘）
$\langle x,y\rangle$	向量或矩阵 x 和 y 的内积
$\mathcal{R}(X)$	矩阵 X 的像空间
$\mathcal{N}(X)$	矩阵 X 的零空间
$\mathrm{Tr}(X)$	矩阵 X 的迹
$\det(X)$	矩阵 X 的行列式
$X \succeq 0$	矩阵 X 半正定
$X \succeq Y$	矩阵 $X-Y$ 半正定
$X \succ 0$	矩阵 X 严格正定
$X \succ Y$	矩阵 $X-Y$ 严格正定
$\|x\|, \|x\|_2$	向量 x 的 ℓ_2 范数
$\|x\|_1$	向量 x 的 ℓ_1 范数（所有分量绝对值的和）
$\|X\|_2$	矩阵 X 的谱范数（最大奇异值）
$\|X\|_1$	矩阵 X 的 ℓ_1 范数（所有分量绝对值的和）
$\|X\|_F$	矩阵 X 的 F 范数
$\|X\|_*$	矩阵 X 的核范数（所有奇异值的和）

符号	含义
$\mathrm{diag}(X)$	方阵 X 所有对角线元素组成的向量
$\mathrm{Diag}(x)$	以向量 x 元素生成的对角矩阵
s^k	在拟牛顿算法中，相邻两次迭代点的差 $x^{k+1}-x^k$
y^k	在拟牛顿算法中，相邻两次迭代点处梯度的差，即 $\nabla f(x^{k+1})-\nabla f(x^k)$
B^k	二阶算法中第 k 步海瑟矩阵的近似矩阵
H^k	二阶算法中第 k 步海瑟矩阵逆的近似矩阵
$\mathcal{E}$	约束优化问题中所有等式约束指标集合
$\mathcal{I}$	约束优化问题中所有不等式约束指标集合
$\mathcal{A}(x)$	约束优化问题中点 x 处的积极集
$P_E(x,\sigma)$	等式约束罚函数
$P_I(x,\sigma)$	不等式约束罚函数
$L(x,\lambda)$	拉格朗日函数
$L_\sigma(x,\lambda)$	增广拉格朗日函数
$\mathrm{prox}_f(x)$	函数 f 的邻近算子
$I_S(x)$	集合 S 的示性函数
$\mathcal{P}_S(x)$	点 x 到闭集 S 的欧几里得投影
$P(A)$	随机事件 A 的概率
$\mathbb{E}[X]$	随机变量 X 的数学期望
$\mathbb{E}[X\|Y]$	随机变量 X 关于 Y 的条件期望

参考文献

[1] ABSIL P A, MAHONY R, SEPULCHRE R. Optimization algorithms on matrix manifolds[M]. Princeton: Princeton University Press, 2009.

[2] ADBY P. Introduction to optimization methods[M]. Berlin: Springer Science & Business Media, 2013.

[3] ARORA R, COTTER A, SREBRO N. Stochastic optimization of PCA with capped MSG[C]//Advances in neural information processing systems. Cambridge, Massachusetts: MIT Press, 2013: 1815-1823.

[4] AVERICK B M, CARTER R G, XUE G L, et al. The MINPACK-2 test problem collection[R]. Chicago: Argonne National Lab, 1992.

[5] BAIRE R, DENJOY A. Leçons sur les fonctions discontinues: professées au collège de france[M]. Paris: Gauthier-Villars, 1905.

[6] BAUSCHKE H H, COMBETTES P L, et al. Convex analysis and monotone operator theory in Hilbert spaces[M]. New York: Springer, 2011.

[7] BECK A, TEBOULLE M. Mirror descent and nonlinear projected subgradient methods for convex optimization[J]. Operations Research Letters, 2003, 31(3): 167-175.

[8] BECK A, TEBOULLE M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems[J]. SIAM Journal on Imaging Sciences, 2009, 2(1): 183-202.

[9] BECK A, TEBOULLE M. Gradient-based algorithms with applications to signal-recovery problems[M]//Convex optimization in signal processing and communications. Cambridge, UK: Cambridge University Press, 2009: 42-88.

[10] BECK A, TETRUASHVILI L. On the convergence of block coordinate descent type methods[J]. SIAM Journal on Optimization, 2013, 23(4): 2037-2060.

[11] BECKER S, BOBIN J, CANDÈS E J. NESTA: a fast and accurate first-order method for sparse recovery[J]. SIAM Journal on Imaging Sciences, 2011, 4(1): 1-39.

[12] BERAHAS A S, BOLLAPRAGADA R, NOCEDAL J. An investigation of Newton-sketch and subsampled Newton methods[J]. Optimization Methods and Software, 2020: 1-20.

[13] BERTSEKAS D P. Constrained optimization and Lagrange multiplier methods[M]. Salt Lake: Academic press, 2014.

[14] BOLTE J, SABACH S, TEBOULLE M. Proximal alternating linearized minimization for nonconvex and nonsmooth problems[J]. Mathematical Programming, 2014, 146(1): 459-494.

[15] BORDES A, BOTTOU L, GALLINARI P. SGD-QN: careful quasi-newton stochastic gradient descent[J]. Journal of Machine Learning Research, 2009, 10(59): 1737-1754.

[16] BORWEIN J, LEWIS A S. Convex analysis and nonlinear optimization: theory and examples[M]. Berlin: Springer Science & Business Media, 2010.

[17] BOTTOU L, CURTIS F E, NOCEDAL J. Optimization methods for large-scale machine learning[J]. SIAM Review, 2018, 60(2): 223-311.

[18] BOUMAL N, MISHRA B, ABSIL P A, et al. Manopt, a MATLAB toolbox for optimization on manifolds[J]. Journal of Machine Learning Research, 2014, 15(42): 1455-1459.

[19] BOYD S, PARIKH N, CHU E, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers[J]. Foundations and Trendső in Machine learning, 2011, 3(1): 1-122.

[20] BOYD S, VANDENBERGHE L. Convex optimization[M]. Cambridge, UK: Cambridge University Press, 2004.

[21] BUNCH J R, PARLETT B N. Direct methods for solving symmetric indefinite systems of linear equations[J]. SIAM Journal on Numerical Analysis, 1971, 8(4): 639-655.

[22] BYRD R H, CHIN G M, NEVEITT W, et al. On the use of stochastic Hessian information in optimization methods for machine learning[J]. SIAM Journal on Optimization, 2011, 21(3): 977-995.

[23] BYRD R H, CHIN G M, NOCEDAL J, et al. Sample size selection in optimization methods for machine learning[J]. Mathematical Programming, 2012, 134(1): 127-155.

[24] BYRD R H, HANSEN S L, NOCEDAL J, et al. A stochastic quasi-Newton method for large-scale optimization[J]. SIAM Journal on Optimization, 2016, 26(2): 1008-1031.

[25] BYRD R H, NOCEDAL J, SCHNABEL R B. Representations of quasi-Newton matrices and their use in limited memory methods[J]. Mathematical Programming, 1994, 63(1-3): 129-156.

[26] BYRD R H, SCHNABEL R B, SHULTZ G A. Approximate solution of the trust region problem by minimization over two-dimensional subspaces[J]. Mathematical Programming, 1988, 40(1-3): 247-263.

[27] CANDÈS E J, LI X, SOLTANOLKOTABI M. Phase retrieval via Wirtinger flow: theory and algorithms[J]. IEEE Transactions on Information Theory, 2015, 61(4): 1985-2007.

[28] CHAMBOLLE A, POCK T. A first-order primal-dual algorithm for convex problems with applications to imaging[J]. Journal of Mathematical Imaging and Vision, 2011, 40(1): 120-145.

[29] CHANG C C, LIN C J. Libsvm: a library for support vector machines[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2011, 2(3): 1-27.

[30] CHEN C, HE B, YE Y, et al. The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent[J]. Mathematical Programming, 2016, 155(1-2): 57-79.

[31] CHEN C, LI M, LIU X, et al. On the convergence of multi-block alternating direction method of multipliers and block coordinate descent method[J]. ArXiv:1508.00193, 2015.

[32] CHEN L, SUN D, TOH K C. A note on the convergence of ADMM for linearly constrained convex optimization problems[J]. Computational Optimization and Applications, 2017, 66(2): 327-343.

[33] DAI Y H, YUAN Y X. A nonlinear conjugate gradient method with a strong global convergence property[J]. SIAM Journal on Optimization, 1999, 10(1): 177-182.

[34] DAI Y, HAN J, LIU G, et al. Convergence properties of nonlinear conjugate gradient methods[J]. SIAM Journal on Optimization, 2000, 10(2): 345-358.

[35] DEFAZIO A, BACH F, LACOSTE-JULIEN S. SAGA: a fast incremental gradient method with support for non-strongly convex composite objectives[C]//Advances in neural information processing systems. Cambridge, Massachusetts: MIT Press, 2014: 1646-1654.

[36] DEMMEL J W. Applied numerical linear algebra[M]. Philadelphia: SIAM, 1997.

[37] DENG W, YIN W. On the global and linear convergence of the generalized alternating direction method of multipliers[J]. Journal of Scientific Computing, 2016, 66(3): 889-916.

[38] DENNIS J E, MEI H. Two new unconstrained optimization algorithms which use function and gradient values[J]. Journal of Optimization Theory and Applications, 1979, 28(4): 453-482.

[39] DENNIS JR J E, SCHNABEL R B. Numerical methods for unconstrained optimization and nonlinear equations[M]. Philadelphia: SIAM, 1996.

[40] DIAMOND S, BOYD S. CVXPY: a Python-embedded modeling language for convex optimization[J]. The Journal of Machine Learning Research, 2016, 17(1): 2909-2913.

[41] DUBOVITSKII A Y, MILYUTIN A A. Extremum problems in the presence of restrictions[J]. USSR Computational Mathematics and Mathematical Physics, 1965, 5(3): 1-80.

[42] FAZEL M, PONG T K, SUN D, et al. Hankel matrix rank minimization with applications to system identification and realization[J]. SIAM Journal on Matrix Analysis and Applications, 2013, 34(3): 946-977.

[43] FEI Y, RONG G, WANG B, et al. Parallel L-BFGS-B algorithm on GPU[J]. Computers & Graphics, 2014, 40: 1-9.

[44] FLETCHER R, REEVES C M. Function minimization by conjugate gradients[J]. The Computer Journal, 1964, 7(2): 149-154.

[45] FLETCHER R. Practical methods of optimization[M]. New Jersey: John Wiley & Sons, 2013.

[46] FLETCHER R, FREEMAN T. A modified Newton method for minimization[J]. Journal of Optimization Theory and Applications, 1977, 23(3): 357-372.

[47] FOURER R, GAY D M, KERNIGHAN B W. A modeling language for mathematical programming[J]. Management Science, 1990, 36(5): 519-554.

[48] FRANK M, WOLFE P. An algorithm for quadratic programming[J]. Naval Research Logistics Quarterly, 1956, 3(1-2): 95-110.

[49] GABAY D, MERCIER B. A dual algorithm for the solution of non linear variational problems via finite element approximation[M]. Institut de recherche d'informatique et d'automatique, 1975.

[50] GHADIMI S, LAN G. Accelerated gradient methods for nonconvex nonlinear and stochastic programming[J]. Mathematical Programming, 2016, 156(1-2): 59-99.

[51] GILBERT J C, NOCEDAL J. Global convergence properties of conjugate gradient methods for optimization[J]. SIAM Journal on Optimization, 1992, 2(1): 21-42.

[52] GILL P E, MURRAY W, WRIGHT M H. Practical optimization[M]. Philadelphia: Society for Industrial, 2019.

[53] GLOWINSKI R, MARROCO A. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires[J]. Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique, 1975, 9(R2): 41-76.

[54] GOEMANS M X, WILLIAMSON D P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming[J]. Journal of the ACM (JACM), 1995, 42(6): 1115-1145.

[55] GOLDFARB D. Curvilinear path steplength algorithms for minimization which use directions of negative curvature[J]. Mathematical Programming, 1980, 18(1): 31-40.

[56] GOLDSTEIN A, PRICE J. An effective algorithm for minimization[J]. Numerische Mathematik, 1967, 10(3): 184-189.

[57] GOODFELLOW I, BENGIO Y, COURVILLE A. Deep learning[M]. Cambridge, Massachusetts: MIT press, 2016.

[58] GOODFELLOW I, POUGET-ABADIE J, MIRZA M, et al. Generative adversarial nets[C]//Advances in neural information processing systems. Cambridge, Massachusetts: MIT Press, 2014: 2672-2680.

[59] GRANT M, BOYD S, YE Y. CVX: MATLAB software for disciplined convex programming[Z]. 2008.

[60] GRIPPO L, LAMPARIELLO F, LUCIDI S. A truncated Newton method with nonmonotone line search for unconstrained optimization[J]. Journal of Optimization Theory and Applications, 1989, 60(3): 401-419.

[61] GRIPPO L, LAMPARIELLO F, LUCIDI S. A nonmonotone line search technique for Newton's method[J]. SIAM Journal on Numerical Analysis, 1986, 23(4): 707-716.

[62] HAGER W W, ZHANG H. A survey of nonlinear conjugate gradient methods[J]. Pacific Journal of Optimization, 2006, 2(1): 35-58.

[63] HALE E T, YIN W, ZHANG Y. Fixed-point continuation for ℓ_1-minimization: methodology and convergence[J]. SIAM Journal on Optimization, 2008, 19(3): 1107-1130.

[64] HAN S P, MANGASARIAN O L. Exact penalty functions in nonlinear programming[J]. Mathematical Programming, 1979, 17(1): 251-269.

[65] HE B, YOU Y, YUAN X. On the convergence of primal-dual hybrid gradient algorithm[J]. SIAM Journal on Imaging Sciences, 2014, 7(4): 2526-2537.

[66] HE B, YUAN X. Convergence analysis of primal-dual algorithms for a saddle-point problem: from contraction perspective[J]. SIAM Journal on Imaging Sciences, 2012, 5(1): 119-149.

[67] HE B, YUAN X. On the $\mathcal{O}(1/n)$ convergence rate of the Douglas–Rachford alternating direction method[J]. SIAM Journal on Numerical Analysis, 2012, 50(2): 700-709.

[68] HESTENES M R, STIEFEL E. Methods of conjugate gradients for solving linear systems[M]. Washington: NBS, 1952.

[69] HIRIART-URRUTY J B, LEMARÉCHAL C. Convex analysis and minimization algorithms i: Fundamentals[M]. Berlin: Springer Science & Business Media, 2013.

[70] HONG M, LUO Z Q. On the linear convergence of the alternating direction method of multipliers[J]. Mathematical Programming, 2017, 162(1-2): 165-199.

[71] HONG M, LUO Z Q, RAZAVIYAYN M. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems[J]. SIAM Journal on Optimization, 2016, 26(1): 337-364.

[72] HU J, JIANG B, LIN L, et al. Structured quasi-Newton methods for optimization with orthogonality constraints[J]. SIAM Journal on Scientific Computing, 2019, 41(4): A2239-A2269.

[73] HU J, MILZAREK A, WEN Z, et al. Adaptive quadratically regularized Newton method for Riemannian optimization[J]. SIAM Journal on Matrix Analysis and Applications, 2018, 39(3): 1181-1207.

[74] JAGGI M. Revisiting Frank-Wolfe: projection-free sparse convex optimization[C]//DASGUPTA S, MCALLESTER D. Proceedings of Machine Learning Research: Proceedings of the 30th international conference on machine learning: vol. 28: 1. Atlanta: PMLR, 2013: 427-435.

[75] JIANG B, MA S, SO A M C, et al. Vector transport-free SVRG with general retraction for Riemannian optimization: complexity analysis and practical implementation[J]. ArXiv:1705.09059, 2017.

[76] JOHNSON R, ZHANG T. Accelerating stochastic gradient descent using predictive variance reduction[C]//Advances in neural information processing systems. Cambridge, Massachusetts: MIT Press, 2013: 315-323.

[77] LEVENBERG K. A method for the solution of certain non-linear problems in least squares[J]. Quarterly of Applied Mathematics, 1944, 2(2): 164-168.

[78] LI X, SUN D, TOH K C. A highly efficient semismooth Newton augmented Lagrangian method for solving Lasso problems[J]. SIAM Journal on Optimization, 2018, 28(1): 433-458.

[79] LI X, SUN D, TOH K C. An asymptotically superlinearly convergent semismooth Newton augmented Lagrangian method for linear programming[J]. ArXiv:1903.09546, 2019.

[80] LI Y, WEN Z, YANG C, et al. A semismooth Newton method for semidefinite programs and its applications in electronic structure calculations[J]. SIAM Journal on Scientific Computing, 2018, 40(6): A4131-A4157.

[81] LIANG S, SRIKANT R. Why deep neural networks for function approximation?[J]. ArXiv:1610.04161, 2016.

[82] LIU D C, NOCEDAL J. On the limited memory BFGS method for large scale optimization[J]. Mathematical Programming, 1989, 45(1-3): 503-528.

[83] LIU J, WRIGHT S J. Asynchronous stochastic coordinate descent: parallelism and convergence properties[J]. SIAM Journal on Optimization, 2015, 25(1): 351-376.

[84] LIU J, WRIGHT S J, RÉ C, et al. An asynchronous parallel stochastic coordinate descent algorithm[J]. The Journal of Machine Learning Research, 2015, 16(1): 285-322.

[85] LOFBERG J. YALMIP: a toolbox for modeling and optimization in MATLAB[C]//Computer aided control systems design, 2004 ieee international symposium on robotics and automation. Washington: IEEE, 2004: 284-289.

[86] LUENBERGER D G, YE Y. Linear and nonlinear programming[M]. 4th ed. Berlin: Springer Publishing Company, Incorporated, 2015.

[87] MCCORMICK G P. A modification of Armijo's step-size rule for negative curvature[J]. Mathematical Programming, 1977, 13(1): 111-115.

[88] MILZAREK A, XIAO X, CEN S, et al. A stochastic semismooth Newton method for nonsmooth nonconvex optimization[J]. SIAM Journal on Optimization, 2019, 29(4): 2916-2948.

[89] MITLIAGKAS I, CARAMANIS C, JAIN P. Memory limited, streaming PCA[C]//Advances in neural information processing systems. Cambridge Massachusetts: MIT Press, 2013: 2886-2894.

[90] MORALES J L, NOCEDAL J. Remark on "algorithm 778: L-BFGS-B: fortran subroutines for large-scale bound constrained optimization" [J]. ACM Transactions on Mathematical Software, 2011, 38(1).

[91] MORDUKHOVICH B S, NAM N M, YEN N. Fréchet subdifferential calculus and optimality conditions in nondifferentiable programming[J]. Optimization, 2006, 55(5-6): 685-708.

[92] MORÉ J J, SORENSEN D C. On the use of directions of negative curvature in a modified Newton method[J]. Mathematical Programming, 1979, 16(1): 1-20.

[93] MORITZ P, NISHIHARA R, JORDAN M. A linearly-convergent stochastic L-BFGS algorithm[C]//Artificial intelligence and statistics. 2016: 249-258.

[94] NASH S G, NOCEDAL J. A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization[J]. SIAM Journal on Optimization, 1991, 1(3): 358-372.

[95] NESTEROV Y. Smooth minimization of non-smooth functions[J]. Mathematical Programming, 2005, 103(1): 127-152.

[96] NESTEROV Y. On an approach to the construction of optimal methods of minimization of smooth convex functions[J]. Ekonomika i Mateaticheskie Metody, 1988, 24(3): 509-517.

[97] NOCEDAL J. Updating quasi-Newton matrices with limited storage[J]. Mathematics of computation, 1980, 35(151): 773-782.

[98] NOCEDAL J, WRIGHT S. Numerical optimization[M]. 2nd ed. Berlin: Springer Science & Business Media, 2006.

[99] OCHS P, CHEN Y, BROX T, et al. iPiano: inertial proximal algorithm for nonconvex optimization[J]. SIAM Journal on Imaging Sciences, 2014, 7(2): 1388-1419.

[100] OUYANG Y, CHEN Y, LAN G, et al. An accelerated linearized alternating direction method of multipliers[J]. SIAM Journal on Imaging Sciences, 2015, 8(1): 644-681.

[101] PAATERO P, TAPPER U. Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values[J]. Environmetrics, 1994, 5(2): 111-126.

[102] PARIKH N, BOYD S, et al. Proximal algorithms[J]. Foundations and Trendső in Optimization, 2014, 1(3): 127-239.

[103] PETERSEN K B, PEDERSEN M S. The matrix cookbook[Z]. Version 20121115. 2012.

[104] PILANCI M, WAINWRIGHT M J. Newton sketch: a near linear-time optimization algorithm with linear-quadratic convergence[J]. SIAM Journal on Optimization, 2017, 27(1): 205-245.

[105] POLAK E, RIBIERE G. Note sur la convergence de méthodes de directions conjuguées[J]. ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique, 1969, 3(R1): 35-43.

[106] POWELL M J. A new algorithm for unconstrained optimization[G] //Nonlinear programming. Amsterdam: Elsevier, 1970: 31-65.

[107] POWELL M. A note on quasi-Newton formulae for sparse second derivative matrices[J]. Mathematical Programming, 1981, 20(1): 144-151.

[108] RECHT B, FAZEL M, PARRILO P. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization[J]. SIAM Review, 2010, 52(3): 471-501.

[109] RICHTÁRIK P, TAKÁ M. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function[J]. Mathematical Programming, 2014, 144(1-2): 1-38.

[110] ROBBINS H, MONRO S, et al. A stochastic approximation method[J]. The Annals of Mathematical Statistics, 1951, 22(3): 400-407.

[111] ROCKAFELLAR R T. Augmented Lagrangians and applications of the proximal point algorithm in convex programming[J]. Mathematics of Operations Research, 1976, 1(2): 97-116.

[112] ROCKAFELLAR R. Convex analysis[M]. Princeton: Princeton University Press, 1970.

[113] RUSZCZYSKI A P, RUSZCZYNSKI A. Nonlinear optimization[M]. Princeton: Princeton University Press, 2006.

[114] SCHMIDT M, LE ROUX N, BACH F. Minimizing finite sums with the stochastic average gradient[J]. Mathematical Programming, 2017, 162(1-2): 83-112.

[115] SHALEV-SHWARTZ S, ZHANG T. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization[C]// International conference on machine learning. 2014: 64-72.

[116] SNYMAN J A. Practical mathematical optimization[M]. Berlin: Springer, 2005.

[117] STEIHAUG T. The conjugate gradient method and trust regions in large scale optimization[J]. SIAM Journal on Numerical Analysis, 1983, 20(3): 626-637.

[118] SU W, BOYD S, CANDÈS E. A differential equation for modeling Nesterov's accelerated gradient method: theory and insights[C]// Advances in neural information processing systems. Cambridge, Massachusetts: MIT Press, 2014: 2510-2518.

[119] SUN R, LUO Z Q, YE Y. On the expected convergence of randomly permuted ADMM[J]. ArXiv:1503.06387, 2015.

[120] SUN W, YUAN Y X. Optimization theory and methods: nonlinear programming[M]. Berlin: Springer Science & Business Media, 2006.

[121] TAHA H A. Operations research: an introduction[M]. New Jersey: Pearson/Prentice Hall, 2011.

[122] TIBSHIRANI R. Regression shrinkage and selection via the Lasso[J]. Journal of the Royal Statistical Society. Series B (Methodological), 1996: 267-288.

[123] TOINT P. A note about sparsity exploiting quasi-Newton updates[J]. Mathematical Programming, 1981, 21(1): 172-181.

[124] UDELL M, MOHAN K, ZENG D, et al. Convex optimization in Julia[C]//Proceedings of the 1st first workshop for high performance technical computing in dynamic languages. 2014: 18-28.

[125] WANG Y X, ZHANG Y J. Nonnegative matrix factorization: A comprehensive review[J]. IEEE Transactions on Knowledge and Data Engineering, 2012, 25(6): 1336-1353.

[126] WEN F, CHU L, LIU P, et al. A survey on nonconvex regularization-based sparse and low-rank recovery in signal processing, statistics, and machine learning[J]. IEEE Access, 2018, 6: 69883-69906.

[127] WEN Z, YIN W. A feasible method for optimization with orthogonality constraints[J]. Mathematical Programming, 2013, 142(1-2): 397-434.

[128] WRIGHT S J. Coordinate descent algorithms[J]. Mathematical Programming, 2015, 151(1): 3-34.

[129] XIAO L, ZHANG T. A proximal stochastic gradient method with progressive variance reduction[J]. SIAM Journal on Optimization, 2014, 24(4): 2057-2075.

[130] XU P, ROOSTA F, MAHONEY M W. Second-order optimization for non-convex machine learning: an empirical study[M]//Proceedings of the 2020 siam international conference on data mining: 199-207.

[131] XU P, YANG J, ROOSTA-KHORASANI F, et al. Sub-sampled Newton methods with non-uniform sampling[C]//Advances in neural information processing systems. Cambridge, Massachusetts: MIT Press, 2016: 3000-3008.

[132] XU Y, YIN W. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion[J]. SIAM Journal on Imaging Sciences, 2013, 6(3): 1758-1789.

[133] YANG L, SUN D, TOH K C. SDPNAL+: a majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints[J]. Mathematical Programming Computation, 2015, 7(3): 331-366.

[134] YIN W, OSHER S, GOLDFARB D, et al. Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing[J]. SIAM Journal on Imaging Sciences, 2008, 1(1): 143-168.

[135] YUAN Y X. On the truncated conjugate gradient method[J]. Mathematical Programming, 2000, 87(3): 561-573.

[136] YUAN Y X. Recent advances in trust region algorithms[J]. Mathematical Programming, 2015, 151(1): 249-281.

[137] ZHANG H, HAGER W W. A nonmonotone line search technique and its application to unconstrained optimization[J]. SIAM Journal on Optimization, 2004, 14(4): 1043-1056.

[138] ZHANG H, REDDI S J, SRA S. Riemannian SVRG: fast stochastic optimization on Riemannian manifolds[C]//Advances in neural information processing systems. Cambridge, Massachusetts: MIT Press, 2016: 4592-4600.

[139] ZHANG J, LIU H, WEN Z, et al. A sparse completely positive relaxation of the modularity maximization for community detection[J]. SIAM Journal on Scientific Computing, 2018, 40(5): A3091-A3120.

[140] ZHANG L, ZHOU W, LI D. Global convergence of a modified Fletcher–Reeves conjugate gradient method with Armijo-type line search[J]. Numerische Mathematik, 2006, 104(4): 561-572.

[141] ZHAO X Y, SUN D, TOH K C. A Newton-CG augmented Lagrangian method for semidefinite programming[J]. SIAM Journal on Optimization, 2010, 20(4): 1737-1765.

[142] ZHU C, BYRD R H, LU P, et al. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization[J]. ACM Transactions on Mathematical Software (TOMS), 1997, 23(4): 550-560.

[143] 徐树方，钱江. 矩阵计算六讲[M]. 北京: 高等教育出版社, 2011.

[144] 徐树方，高立，张平文. 数值线性代数[M]. 北京: 北京大学出版社, 2013.

[145] 袁亚湘，孙文瑜. 最优化理论与方法[M]. 北京: 科学出版社, 2011.

索引

Symbols

α-下水平集 31
ℓ_1 罚函数 199

A

AdaDelta 295
AdaGrad 293
Adam 295
ADMM ..... 见交替方向乘子法
鞍点 83
凹函数 40

B

BB 步长 132
BB 方法 131, 157
BP 问题 ..... 见基追踪问题
半定规划 68
　～ 的对偶 96
半空间 36
半正定锥 37
闭函数 32
步长 118

C

Chambolle-Pock 算法 264
continuation ..... 见连续化
超平面 36
次梯度 49
次微分 49

D

Douglas-Rachford Splitting 算法 273
Dubovitskii-Milyutin 定理 53
大残量问题 172
单调映射 42
动量方法 291
对偶变量 ..... 见拉格朗日乘子
对偶间隙 89
对偶近似点梯度法 258
对偶可行解 89
对偶锥 91
对数罚函数 197
多层感知机 7, 298
多面体 37

E

二次罚函数
　不等式约束的 ～ 194
　等式约束的 ～ 188
　一般约束的 ～ 194
二次共轭函数 49

二次上界 26
二次下界 46
二次约束二次规划问题 69

F

Farkas 引理 102
Fenchel 不等式 47
Fréchet 可微 24, 28
Frobenius 内积 23
罚函数法 187
范数
ℓ_p ~ 21
~ 的相容性 23
Frobenius ~ 22
核 ~ 23
矩阵 2 ~ 23
矩阵 ~ 22
算子 ~ 22
向量 ~ 21
方差减小技术 301
仿射包 34
仿射集 33
非负矩阵分解 72
分布式鲁棒优化 67
分块坐标下降法 252, 254
分离超平面定理 38
分组 LASSO 251
复合优化问题 64, 219
~ 的一阶必要条件 86

G

Gâteaux 可微 28
高斯 – 牛顿算法 173
割线方程 150
共轭函数 47
广义不等式 90
广义实值函数 30

H

Huber 损失函数 134
海瑟矩阵 25
互补松弛条件 103
回退法 120

J

基追踪问题 4, 60, 195, 211
积极集 99
几何最优性条件 98
交替方向乘子法 270
多块 ~ 279
线性化 ~ 277
交替极小化方法 260
截断共轭梯度法 167
近似点算法 242
近似点梯度法 224
精确罚函数 199
局部极小解 14
矩阵内积 ..... 见 Frobenius 内积
矩阵优化 71
卷积神经网络 8
决策变量 1

K

KKT 对 103
KKT 条件
凸优化问题的 ~ 108
一般约束优化问题的 ~ ..103
柯西不等式 22, 24, 50
柯西点 167

可行点 ..... 见可行解
可行解 1
可行域 1

L

L-BFGS 方法 155, 158
～的双循环递归算法 156
LASSO 问题 6
LASSO 问题求解
～ 的 Chambolle-Pock 算法 265
～的 Nesterov 算法 237
～的 PDHG 算法 265
～的次梯度算法 141
～的分块坐标下降法 255
～的交替方向乘子法 280, 282
～的近似点算法 246
～的近似点梯度法 226
～的梯度法 133
Levenberg-Marquardt 方法 ... 见 LM 方法
LICQ ..... 见线性无关约束品性
LM 方法 176
LMF 方法 179
信赖域型～ 176
拉格朗日乘子 88
拉格朗日对偶函数 88
拉格朗日对偶问题 89
拉格朗日函数 88
广义不等式约束优化问题的～ 92
连续化 190, 197
临界锥 104
邻近算子 220

M

MFCQ 101
Momentum ..... 见动量方法
Moreau 分解 259
Moreau-Rockafellar 定理 53
目标函数 1

N

Nesterov 算法
FISTA 算法 231
～的等价变形 232
～的线搜索 233
第二类～ 235
第三类～ 236
非凸问题的～ 236
随机优化问题的～ 292
内点罚函数 197
拟牛顿格式
BFGS 公式 153
DFP 公式 154
SR1 公式 152
牛顿法
～的收敛性 143, 146
非精确～ 146
经典～ 142
修正～ 145
牛顿方程 142
牛顿方向 142

P

PDHG 算法 见原始对偶混合梯度法

Q

QR 分解 177

前馈神经网络 ..... 见多层感知机
强对偶原理 89, 107
强凸函数 40
～的等价定义 40
强凸参数 40
切锥 97
球
范数～ 37
欧几里得～ 36
曲率条件 150
全局极小解 14

R

ReLU 函数 8, 64
RMSProp 294
弱对偶原理 89

S

SAG 方法 303
SAGA 方法 303
Sherman-Morrison-Woodbury 公式 153
Sigmoid 函数 8
Slater 约束品性 107
Steihaug-CG ．见截断共轭梯度法
SVRG 方法 306
上方图 31
深度学习 7
示性函数 48, 223
适当函数 30
适当锥 90
收敛速度
Q-～ 18
R-～ 19
收敛准则 16
数据插值 63
数据拟合 61
搜索方向 118
算法收敛性
全局依点列收敛 17
依点列收敛 17
依函数值收敛 17
随机梯度下降算法 290
随机优化问题 66
随机主成分分析 67

T

泰勒展开 25
梯度 24, 28
梯度利普希茨连续 26
梯度映射 228
停机准则 16
凸包 34
凸函数 39
凸集 33
凸组合 34
图像处理模型
TV-L^1 模型 265
盲反卷积模型 65
图像去噪 65
椭球 36

W

Weierstrass 定理 79, 221
外点罚函数 188
稳定点 82
无约束优化最优性条件
二阶必要条件 82

二阶充分条件 83
一阶必要条件 81

X

稀疏优化 3
下半连续函数 32
下降方向 81, 118
线搜索准则
Armijo 准则 119
Goldstein 准则 121
Wolfe 准则 122
非单调 ~ 122
线性规划 2, 59
~ 的对偶 93
~ 的 KKT 条件 110
标准形 ~ 59
不等式形 ~ 60
线性化可行方向锥 99
线性无关约束品性 101
线性约束品性 102
相对内点 107
相位恢复 179
小残量问题 172
信赖域 160
信赖域半径 160
信赖域算法 161

Y

压缩感知 3
雅可比矩阵 25
严格分离定理 38
严格互补松弛条件 103
严格局部极小解 14
严格凸函数 40
有限内存 BFGS 方法 . 见 L-BFGS 方法
余强制性 129
原始对偶混合梯度算法 263
约束集合 ..... 见可行域
约束品性 97
约束优化最优性条件
二阶必要条件 104
二阶充分条件 105
一阶最优性条件 见 KKT 条件

Z

Zoutendijk 条件 125
增广拉格朗日函数 200, 206
增广拉格朗日函数法 200
等式约束优化问题的 ~ .. 200
凸问题的 ~ 208
一般约束优化问题的 ~ .. 206
支撑超平面 39
支撑超平面定理 39
支撑函数 48
锥
二次 ~ 37
范数 ~ 37
凸 ~ 35
锥组合 35
自对偶锥 92
最大割问题
~ 的凸松弛 71
~ 的原始形式 70
最小二乘问题 61
非线性 ~ 61, 172
线性 ~ 61, 62