1.5em 0pt

线性代数进阶课程

Advanced Linear Algebra Course

姓　　名：姓名

学　　号：学号

学　　院：计算机科学与技术学院

专　　业：计算机科学与技术

课任教师：方发明

2022　年　07　月

目录

摘要

在计算机广为普及的今天，最优化方法已然成为工程技术人员研究大规模优化问题所必备的研究工具。凸优化，作为最优化的一个子领域，研究定义于凸集中的凸函数最小化的问题。由于近年来计算机算力的提高和最优化理论的充分发展，凸优化问题以其优秀的性质在机器学习领域中拥有广泛的应用。本文针对凸优化模型展开研究，首先罗列了一些基础的数学概念，并对于所涉及的定理给予了详细的证明。然后分别介绍了牛顿迭代法、拟牛顿方法、梯度下降法、随机梯度类方法、共轭梯度法、临近梯度下降法、增广拉格朗日法和交替方向乘子法这八种优化方法。在每个方法中，首先描述了该方法适用的凸优化模型，然后介绍了其对应的算法流程和具体的迭代格式，接着基于方法进行收敛性或优缺点分析，最后列举了如 Lasso 问题、最小二乘问题、矩阵分解问题等经典模型，以具体的应用实例来增加读者对于方法在实际机器学习问题中使用的理解。

关键词: 凸优化，机器学习，牛顿法，梯度法，乘子法

Abstract

With the popularization of computers, optimization method has become a necessary research tool for engineers to resolve large-scale optimization problems. As a subfield of optimization, convex optimization studies the minimization of convex functions defined in convex sets. Because of the improvement of computing power and the full development of optimization theory, convex optimization problem has been widely used in machine learning regarding its excellent properties. In this paper, the convex optimization model is studied. Firstly, some essential mathematical concepts are listed, and the theorems involved are proved in detail. Then, eight optimization methods including Newton-Raphson method, Quasi-Newton method, Gradient descent method, Stochastic gradient descent method, Conjugate gradient method, Proximal gradient method, Augmented Lagrange method, and Alternating direction method of multipliers, are introduced respectively. In each scenario, the convex optimization model applicable to the method is described first, then the corresponding algorithm flow and specific iterative format are presented. Furthermore, the convergence or advantages and disadvantages of the method are analyzed. Finally, classical models such as the Lasso problem, the least square problem, and the matrix decomposition problem are listed. Concrete application examples are presented to increase the reader's understanding of the use of methods in real machine learning problems.

Keywords: Convex Optimization, Machine Learning, Newton's Method, Gradient Method, Multiplier Method

1. 引言

凸优化问题是在凸集 $\mathcal{X}$ 上求解目标凸函数 $f(\cdot)$ 最小值的一类最优化问题。一般这类问题可以用于求解如何分配资源使收益达到最优，在对问题进行定性和定量的分析后，将资源与收益的关系抽象成适当的数学模型，作为目标函数 $f(\cdot)$，根据模型的不同设计不同的优化方法进行求解。也就是对于不同的目标函数 $f(\cdot)$，可以使用不同的优化方法求解。

凸优化问题的求解在许多科学与工程领域有广泛的应用，例如在数据分析与机器学习、金融与经济、工业生产等方面都有具体应用。

根据凸优化问题是否有约束，或根据目标函数的形式不同，分别有牛顿类方法与梯度类方法求解无约束优化问题，特别地，对于目标函数存在不可微的情况，可以在梯度概念的基础上，引入临近梯度，适用临近梯度下降法求解；而对于有约束的优化问题，可以通过引入乘子，适用乘子法进行求解。

2. 数学概念

2.1 适当函数

定义 2.1. 给定函数 $f(\cdot)$ 和非空集合 $\mathcal{X}$，如果存在 $x \in \mathcal{X}$ 使得 $f(x) < +\infty$，并且对 $\forall xin\mathcal{X}$，都有 $f(x) > -\infty$，那么称函数 $f(\cdot)$ 关于集合 $\mathcal{X}$ 是适当的

由定义可知，适当函数 $f(\cdot)$ 具有在某个非空集合 $\mathcal{X}$ 上至少有一处取值不为正无穷，并且处处不为负无穷的特点。

此外，规定适当函数 $f(\cdot)$ 的定义域为 $\text{dom}(f) = \{x|f(x) < +\infty\}$.

2.2 凸函数

首先引入凸集的概念，直观来看凸集就是任取集合中的两点，其构成的线段都在集合内就称为凸集，下面给出具体的定义。

定义 2.2. 给定集合 C，对 $\forall x_1, x_2 \in C$，如果对 $0 \leq \theta \leq 1$，都有 $\theta x_1 + (1-\theta)x_2 \in C$，则称这样的集合 C 为凸集

例如，超平面 $\{\boldsymbol{x}|\boldsymbol{a^T x} = \boldsymbol{b}\}$ 是典型的凸集，其中 $\boldsymbol{a}$ 为任意的非零向量。

问题 2.3. 超平面 $H = \{\boldsymbol{x}|\boldsymbol{a^T x} = \boldsymbol{b}\}$ 是凸集。

证明: 任取 $\boldsymbol{x_1}, \boldsymbol{x_2} \in H, 0 \leq \theta \leq 1$，有

$$\begin{cases} \boldsymbol{a^T x_1} = \boldsymbol{b}, \\ \boldsymbol{a^T x_2} = \boldsymbol{b}. \end{cases}$$

由 $0 \leq \theta \leq 1$，进一步可得，

$$\begin{cases} \theta\boldsymbol{a^T x_1} = \theta\boldsymbol{b}, \\ (1-\theta)\boldsymbol{a^T x_2} = (1-\theta)\boldsymbol{b}. \end{cases}$$

将上述两式相加可得，

$$\theta\boldsymbol{a^T x_1} + (1-\theta)\boldsymbol{a^T x_2} = \theta\boldsymbol{b} + (1-\theta)\boldsymbol{b}$$

$$\boldsymbol{a^T}(\theta\boldsymbol{x_1} + (1-\theta)\boldsymbol{x_2}) = \boldsymbol{b}$$

因此，$\theta\boldsymbol{x_1} + (1-\theta)\boldsymbol{x_2} \in H$，即超平面 H 是凸集。 □

有了凸集的概念，下面给出凸函数的定义。

定义 2.4. 设函数 $f(\cdot)$ 是适当函数，如果 $\mathrm{dom}(f)$ 是凸集，且对 $\forall x, y \in \mathrm{dom}(f), 0 \leq \theta \leq 1$，都有

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y), \tag{2.1}$$

则称这样的 $f(\cdot)$ 为凸函数；如果同时还满足 $\forall x, y \in \mathrm{dom}(f), x \neq y, 0 < \theta < 1$，则称这样的 $f(\cdot)$ 为严格凸函数。

直观来看，连接凸函数图像上的任意两点，其构成的线段都在图像上方。如果连接凸向上的任意两点，其构成的线段都在图像下方，则这样的函数称为凹函数。

例如，负熵函数 $f(x) = x \ln x, x > 0$ 是凸函数，其函数图像如**??**所示。

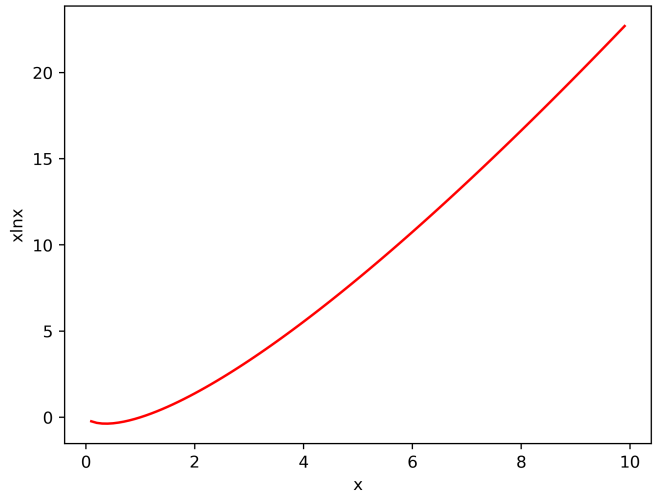

图 2-1: $f(x) = x \ln x$ 函数图像

更进一步，可以定义强凸函数的概念。

定义 2.5. 若存在常数 $m > 0$，使得

$$g(x) = f(x) - \frac{m}{2}\|x\|_2^2 \tag{2.2}$$

为凸函数，则称这样的 $f(\cdot)$ 为强凸函数或 m-强凸函数，m 称为强凸参数。

由上述的定义，$g(x)$ 是一个凸函数，因此 $g(x)$ 根据凸函数的定义可以得到如下的等价定义。

定义 2.6. 若存在常数 $m > 0$，使得对 $\forall x, y \in \mathrm{dom}(f), 0 < \theta < 1$，有

$$f(\theta x + (1-\theta y)) \leq \theta f(x) + (1-\theta) f(y) - \frac{m}{2}\theta(1-\theta)\|x-y\|_2^2 \tag{2.3}$$

则称这样的 $f(\cdot)$ 为强凸函数或 m-强凸函数，m 称为强凸参数。

下面给出上述两个定义是等价的证明。

证明: 由定义**??**可知，$g(x)$ 为凸函数，于是有任意的 $x, y \in \mathbb{R}, 0 < \theta < 1$，

$$g(\theta x + (1-\theta)y) \leq \theta g(x) + (1-\theta)g(y)$$

$$f(\theta x + (1-\theta)y) - \frac{m}{2}\|\theta x + (1-\theta)y\|_2^2 \leq \theta(f(x) - \frac{m}{2}\|x\|_2^2) + (1-\theta)(f(y) - \frac{m}{2}\|y\|_2^2)$$

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y) + \frac{m}{2}\theta\|x\|_2^2 - fracm2(1-\theta)\|y\|_2^2 + \frac{m}{2}\|\theta x$$

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y) + \frac{m}{2}\theta(1-\theta)\|x-y\|_2^2$$

与定义**??**的形式相同，因此定义**??**和定义**??**等价。 □

显然，当强凸函数的强凸参数 $m = 0$ 时，有 $g(x) = f(x)$，即 $f(x)$ 退化成了一个凸函数。此外，根据等价定义，也可以得到 $f(x)$ 同时是一个严格凸函数。因此，强凸函数一定是严格凸函数，当 $m = 0$ 时退化成凸函数。

下面考虑强凸函数关于最小值的定理。

定理 2.7. 设 $f(\cdot)$ 是强凸函数，且 $f(\cdot)$ 存在最小值，则最小值点唯一。

证明: 假设 $f(\cdot)$ 存在两个不同的点 $x \neq y, x, y \in \text{dom}(f)$ 都是 $f(\cdot)$ 的最小值点，

那么由强凸函数的等价定义，对 $\forall 0 < \theta < 1, m > 0$，有

$$\begin{aligned} f(\theta x + (1-\theta)y) &\leq \theta f(x) + (1-\theta)f(y) - \frac{m}{2}\theta(1-\theta)\|x-y\|_2^2 \\ &= f(x) - \frac{m}{2}\theta(1-\theta)\|x-y\|_2^2 \\ &< f(x) \end{aligned}$$

与假设 $f(x)$ 为最小值矛盾，因此假设不成立，即如果 $f(\cdot)$ 是强凸函数，且 $f(\cdot)$ 存在最小值，则最小值点唯一。 □

最后分析凸函数的一些基本性质。

定理 2.8. 若 f_1, f_2 是凸函数，则 $f_1 + f_2$ 也是凸函数

证明: 由 f_1, f_2 是凸函数可知，对 $\forall x, y \in \text{dom}(f), 0 \leq \theta \leq 1$ 有

$$\begin{aligned} f_1(\theta x+(1-\theta)y)+f_2(\theta x+(1-\theta)y) &\le \theta f_1(x)+(1-\theta)f_1(y)+\theta f_2(x)+(1-\theta)f_2(y) \\ &= \theta(f_1(x)+f_2(x))+(1-\theta)(f_1(y)+f_2(y)) \end{aligned}$$

由凸函数的定义，f_1+f_2 是凸函数。 □

定理 2.9. 若 $f_1, f_2, \cdots, f_m$ 是凸函数，则 $f(x)=\max\{f_1, f_2, \cdots, f_m\}$ 也是凸函数

证明: 对 $\forall x, y \in \mathrm{dom}(f), 0 \le \theta \le 1$，有

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

注意到 $f(x)=\max\{f_1, f_2, \cdots, f_m\}$，因此 $f(x) \le f_i(x), i=1,2,\cdots,m$，于是可以得到，

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

由凸函数的定义，$f(x)=\max\{f_1, f_2, \cdots, f_m\}$ 是凸函数。 □

定义 2.10 (聚点). 任给 $\varepsilon>0$，存在无穷多个 z_n 满足

$$|z_n - z| < \varepsilon.$$

则称 z 为复数序列 $\{z_n\}$ 的一个聚点。

定义 2.11 (向量内积). 给定向量 $\boldsymbol{x} \in \mathbb{R}^n$ 和 $\boldsymbol{y} \in \mathbb{R}^n$, 定义两个向量的内积为

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{n} x_i y_i,$$

如果 $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$, 则记为 $\boldsymbol{x} \perp \boldsymbol{y}$。

定理 2.12 (柯西-施瓦茨不等式). 对任意的 $\boldsymbol{x} \in \mathbb{R}^n$ 和 $\boldsymbol{y} \in \mathbb{R}^n$ 有

$$|\langle \boldsymbol{x}, \boldsymbol{y} \rangle| \le \|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\|,$$

该不等式称为柯西-施瓦茨不等式 (Cauchy-Schwarz inequality)[2007 柯西]。

证明: 设 $x=(x_1,x_2...x_n), y=(y_1,y_2...y_n)$，则

$$|\langle \boldsymbol{x},\boldsymbol{y}\rangle|^2=(\boldsymbol{x}_1\boldsymbol{y}_1+...+\boldsymbol{x}_n\boldsymbol{y}_n)^2, \|\boldsymbol{x}\|^2\cdot\|\boldsymbol{y}\|^2=(\boldsymbol{x}_1^2+...+\boldsymbol{x}_n^2)\cdot(\boldsymbol{y}_1^2+...+\boldsymbol{y}_n^2),$$

构造方程

$$(\boldsymbol{x}_1\boldsymbol{z}-\boldsymbol{y}_1)^2+...+(\boldsymbol{x}_n\boldsymbol{z}-\boldsymbol{y}_n)^2=0,$$

其中 $\boldsymbol{z}\in\mathbb{R}^n$ 且为未知数。将方程拆分成以下形式

$$(\boldsymbol{x}_1^2+...+\boldsymbol{x}_n^2)\boldsymbol{z}^2-2(\boldsymbol{x}_1\boldsymbol{y}_1+...+\boldsymbol{x}_n\boldsymbol{y}_n)\boldsymbol{z}+(\boldsymbol{y}_1^2+...+\boldsymbol{y}_n^2)=0,$$

因为方程参数固定，只有一个未知数，因此至多只有一个解，即 $\Delta\leq 0$，也即

$$\Delta=4(\boldsymbol{x}_1\boldsymbol{y}_1+...+\boldsymbol{x}_n\boldsymbol{y}_n)^2-4(\boldsymbol{x}_1^2+...+\boldsymbol{x}_n^2)(\boldsymbol{y}_1^2+...+\boldsymbol{y}_n^2)\leq 0,$$

故

$$|\langle \boldsymbol{x},\boldsymbol{y}\rangle|\leq\|\boldsymbol{x}\|\cdot\|\boldsymbol{y}\|.$$

不等式成立。 □

2.3 梯度与次梯度

定义 2.13 (梯度). 给定函数 $f:\mathbb{R}^n\to\mathbb{R}$ 在 $\boldsymbol{x}\in\mathbb{R}^n$ 的梯度定义为:

$$\nabla f(\boldsymbol{x}):=\begin{pmatrix}\frac{\partial f(\boldsymbol{x})}{\partial x_1}\\ \vdots\\ \frac{\partial f(x)}{\partial x_n}\end{pmatrix},$$

其中 $\frac{\partial f(x)}{\partial x_i}:=\lim_{t\to 0}\frac{f(x+te_i)-f(x)}{t}$ ， e_i 为第 $\boldsymbol{i}$ 个元素为 1 的单位向量 $(0,\cdots,1,\cdots,0)^\top$ 。

定义 2.14 (次梯度). 函数 $f:\mathbb{R}^n\to\mathbb{R}$ 是一个适当的函数且对于 $\boldsymbol{x}\in\mathrm{dom}(f)$, 如果向量 $\boldsymbol{g}\in\mathbb{R}^n$ 满足

$$f(\boldsymbol{y})\geq f(\boldsymbol{x})+\langle \boldsymbol{g},\boldsymbol{y}-\boldsymbol{x}\rangle, \forall\boldsymbol{y}\in\mathbb{R}^n, \boldsymbol{y}\in\mathrm{dom}(f),$$

则称 $\boldsymbol{g}$ 为函数 $f(\cdot)$ 在 $\boldsymbol{x}$ 处的一个次梯度 (sub-gradient)。

定义 2.15 (次微分). 函数 $f:\mathbb{R}^n\to\mathbb{R}$ 在 $\boldsymbol{x}\in\mathrm{dom}(f)$ 的所有次梯度的集合称为函数 $f(\cdot)$ 在

$\boldsymbol{x}$ 的次微分 (sub-differential)，记为 $\partial f(\boldsymbol{x})$：

$$\partial f(\boldsymbol{x}) := \{\boldsymbol{g} \in \mathbb{R}^n \mid f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \boldsymbol{g}, \boldsymbol{y} - \boldsymbol{x} \rangle, \forall \boldsymbol{y} \in \text{dom}(f)\}\,.$$

2.4 海塞矩阵

海塞矩阵 (Hesse 矩阵)，是一个多元函数的二阶偏导数构成的方阵，常用于牛顿法解决优化问题：若一元函数 $f(x)$ 在 $x = x_0$ 点的某个邻域内具有任意阶导数，则 $f(x)$ 在 x_0 点处的泰勒展开式为：

$$f(x) = f(x_0) + f'(x_0)\Delta x + \frac{1}{2}f''(x_0)(\Delta x)^2 + ... \tag{2.4}$$

其中 $\Delta x = x - x_0$, $\Delta x^2 = (x - x_0)^2$

二元函数 $f(x_1, x_2)$ 在 $X_0(x_0, y_0)$ 点处的泰勒展开式为：

$$f(x,y) = f(x_0,y_0) + \frac{\partial f}{\partial x}\bigg|_{X_0}\Delta x_0 + \frac{\partial f}{\partial y}\bigg|_{X_0}\Delta y_0 + \frac{1}{2}\left[\frac{\partial^2 f}{\partial x^2}\bigg|_{X_0}\Delta x_0^2 + 2\,\frac{\partial^2 f}{\partial x \partial y}\bigg|_{X_0}\Delta x_0 \Delta y_0 + \frac{\partial^2 f}{\partial y^2}\bigg|_{X_0}\Delta y_0^2\right] + ... \tag{2.5}$$

其中 $\Delta x = x - x_0$,$\Delta y = y - y_0$. 将上述展开式写成矩阵形式，则有：

$$f(X) = f(X_0) + (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y})\bigg|_{X_0}\begin{pmatrix}\Delta x \\ \Delta y\end{pmatrix} + \frac{1}{2}(\Delta x, \Delta y)\begin{pmatrix}\frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2}\end{pmatrix}\bigg|_{X_0}\begin{pmatrix}\Delta x \\ \Delta y\end{pmatrix} \tag{2.6}$$

即：

$$f(X) = f(X_0) + \nabla f(X_0)^{\mathbb{T}}\Delta X + \frac{1}{2}X^{\mathbb{T}}G(X_0)\Delta X + ... \quad . \tag{2.7}$$

其中：

$$G(X_0) = \begin{pmatrix}\frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2}\end{pmatrix}\bigg|_{X_0}, \Delta X = \begin{pmatrix}\Delta x \\ \Delta y\end{pmatrix}. \tag{2.8}$$

$G(X_0)$ 是 $f(x,y)$ 在 X_0 点处的 Hesse 矩阵。它是由函 $f(x,y)$ 在 X_0 点处的二阶偏导数所组成的方阵。将二元函数的泰勒展开式推广到多元函数，则 $f(x_1, x_2..., x_n)$ 在 X_0 点处的泰勒展开式的矩阵形式为：

$$f(X) = f(X_0) + \nabla f(X_0)^{T}\Delta X + \frac{1}{2}X^{T}G(X_0)\Delta X + ... \quad . \tag{2.9}$$

其中:

(1) $\nabla f(X_0) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}\right]\Big|_{X_0}^{T}$，它是 $f(X)$ 在 X_0 点处的梯度。

(2) $G(X_0) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}\Bigg|_{X_0}$ 为 $f(X)$ 在 X_0 点处的 Hesse 矩阵。

定义 2.16 (Hesse 矩阵). Hesse 矩阵是由目标函数 $f(\cdot)$ 在点 X 处的二阶偏导数组成的 $n \times n$ 阶对称矩阵。

2.5 李普希茨连续可微与 $L-$ 光滑性质

定义 2.17 (李普希茨连续). 如果函数 $f : \mathbb{R}^n \to \mathbb{R}$ 是适当的凸函数，且满足

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|, \forall \boldsymbol{x}, \boldsymbol{y} \in \mathrm{dom}(f),$$

则我们认为函数 $f(\cdot)$ 是李普希茨 (Lipschitz) 连续的，其中常数 L 称为光滑参数。

定义 2.18 ($L-$ 光滑). 给定 $L \geq 0$, 函数 $f : \mathbb{R}^n \to \mathbb{R}$ 定义在集合 $\mathcal{D}$ 上是可微的，且满足

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{D},$$

则该函数 $f(\cdot)$ 是 $L-$ 光滑的，而常数 L 称为光滑参数。

问题 2.19 (二次函数的 $L-$ 光滑). 考虑二次函数 $f : \mathbb{R}^n \to \mathbb{R}$ 是

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^\top \boldsymbol{x} + c,$$

其中 $\boldsymbol{A} \in \mathbb{R}^n$ 为对称矩阵且 $c \in \mathbb{R}$。证明 $\|\boldsymbol{A}\|_{p,q}$ 是最小的光滑参数，其中 $\|\cdot\|_{p,q}$ 表示诱导范数，即

$$\|\boldsymbol{A}\|_{p,q} = \max\left\{\|\boldsymbol{A}\boldsymbol{x}\|_q \mid \|\boldsymbol{x}\|_p \leq 1\right\},$$

要求 $q \in [1, \infty)$ 且 $\frac{1}{p} + \frac{1}{q} = 1$。

解: 考虑定义在 $\mathbb{R}^n$ 上的一般 l_p- 范数 $(1 \leq p \leq \infty)$。那么对任意的 $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$,

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_q = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}\boldsymbol{y}\|_q \leq \|\boldsymbol{A}\|_{p,q}\|\boldsymbol{x} - \boldsymbol{y}\|_p,$$

依题意，

$$\|\boldsymbol{A}\|_{p,q} = \max\left\{\|\boldsymbol{A}\boldsymbol{x}\|_q \mid \|\boldsymbol{x}\|_p \leq 1\right\},$$

我们可以说明函数 $f(\cdot)$ 是 $\|\boldsymbol{A}\|_{p,q}$ 光滑的。特别需要说明的是，$\|\boldsymbol{A}\|_{p,q}$ 是最小的光滑参数。加入函数 $f(\cdot)$ 是 $L-$ 光滑的，那么取一个 $\tilde{\boldsymbol{x}}$ 满足 $\|\tilde{\boldsymbol{x}}\|_p = 1$ 和 $\|\boldsymbol{A}\tilde{\boldsymbol{x}}\|_q = \|\boldsymbol{A}\|_{p,q}$，那么

$$\|\boldsymbol{A}\|_{p,q} = \|\boldsymbol{A}\tilde{\boldsymbol{x}}\|_q = \|\nabla f(\tilde{\boldsymbol{x}}) - \nabla f(\boldsymbol{0})\|_q \leq L\|\tilde{\boldsymbol{x}} - \boldsymbol{0}\|_p = L,$$

从而说明 $\|\boldsymbol{A}\|_{p,q}$ 是最小的光滑参数。 ■

下面介绍关于 L-光滑函数的非常有用的结论，即下降引理（Descent Lemma），该引理所表达的是该函数有一个特定的二次函数上界。

定理 2.20 (下降引理). 如果函数 $f : \mathbb{R}^n \to \mathbb{R}$ 是一个定义在给定凸集 D 上的 L-光滑函数，那么对于 $\forall \boldsymbol{x}, \boldsymbol{y} \in D$，有

$$f(y) \leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \tag{2.10}$$

证明: 由微积分基本定理，

$$f(\boldsymbol{y}) - f(\boldsymbol{x}) = \int_0^1 \langle \nabla f(\boldsymbol{x} + t(\boldsymbol{y} - \boldsymbol{x})), \boldsymbol{y} - \boldsymbol{x}\rangle dt$$

因此，

$$f(\boldsymbol{y}) - f(\boldsymbol{x}) = \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle + \int_0^1 \langle \nabla f(x + t(\boldsymbol{y} - \boldsymbol{x})) - \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle dt$$

进一步可以得到，

$$\begin{aligned} |f(\boldsymbol{y}) - f(\boldsymbol{x}) - \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle| &= |\int_0^1 \langle \nabla f(x + t(\boldsymbol{y} - \boldsymbol{x})) - \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle dt| \\ &\leq \int_0^1 |\langle \nabla f(x + t(\boldsymbol{y} - \boldsymbol{x})) - \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle| dt \end{aligned}$$

由 Cauchy-Schwarz 不等式可得，

$$\begin{aligned}|f(\boldsymbol{y})-f(\boldsymbol{x})-\langle\nabla f(\boldsymbol{x}),\boldsymbol{y}-\boldsymbol{x}\rangle| &\leq \|\nabla f(x+t(\boldsymbol{y}-\boldsymbol{x}))-\nabla f(\boldsymbol{x})\|_2\cdot\|\boldsymbol{y}-\boldsymbol{x}\|_2 dt\\ &\leq \int_0^1 tL\|\boldsymbol{y}-\boldsymbol{x}\|_2^2 dt\\ &= \frac{L}{2}\|\boldsymbol{y}-\boldsymbol{x}\|_2^2\end{aligned}$$

整理可得，

$$f(\boldsymbol{y})\leq f(\boldsymbol{x})+\langle\nabla f(\boldsymbol{x}),\boldsymbol{y}-\boldsymbol{x}\rangle+\frac{L}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2$$

定理 2.21 ($L-$ 光滑的多种刻画性质). 如果函数 $f:\mathbb{R}^n\to\mathbb{R}$ 是一个连续可微凸函数，那么下面的性质均是等价的：

i). $f(\cdot)$ 是 $L-$ 光滑的;

ii). 对任意的 $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^n$，

$$f(\boldsymbol{y})\leq f(\boldsymbol{x})+\langle\nabla f(\boldsymbol{x}),\boldsymbol{y}-\boldsymbol{x}\rangle+\frac{L}{2}\|\boldsymbol{x}-\boldsymbol{y}\|^2.$$

iii). 对任意的 $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^n$，

$$f(\boldsymbol{y})\geq f(\boldsymbol{x})+\langle\nabla f(\boldsymbol{y}),\boldsymbol{y}-\boldsymbol{x}\rangle+\frac{1}{2L}\|\nabla f(\boldsymbol{x})-\nabla f(\boldsymbol{y})\|^2.$$

iv). 对任意的 $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^n$，

$$\langle\nabla f(\boldsymbol{x})-\nabla f(\boldsymbol{y}),\boldsymbol{x}-\boldsymbol{y}\rangle\geq\frac{1}{L}\|\nabla f(\boldsymbol{x})-\nabla f(\boldsymbol{y})\|^2.$$

v). 对任意的 $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^n$ 和 $\lambda\in[0,1]$，

$$f(\lambda\boldsymbol{x}+(1-\lambda)\boldsymbol{y})\geq\lambda f(\boldsymbol{x})+(1-\lambda)f(\boldsymbol{y})-\frac{L}{2}\lambda(1-\lambda)\|\boldsymbol{x}-\boldsymbol{y}\|^2.$$

证明: i)⇒ii)：由下降引理得到；

ii)⇒iii)：假设 $\nabla f(\boldsymbol{x})=\nabla f(\boldsymbol{y})$，对于给定的 $\boldsymbol{x}\in\mathbb{R}^n$，考虑

$$g_x(\boldsymbol{y})=f(\boldsymbol{y})-f(\boldsymbol{x})-\langle\nabla f(\boldsymbol{x}),\boldsymbol{y}-\boldsymbol{x}\rangle,\boldsymbol{y}\in\mathbb{R}^n,$$

对于该函数 $g_x(\cdot)$，有

$$\begin{aligned} g_{\boldsymbol{x}}(\boldsymbol{z}) &= f(\boldsymbol{z}) - f(\boldsymbol{x}) - \langle \nabla f(\boldsymbol{x}), \boldsymbol{z} - \boldsymbol{x} \rangle \\ &\leq f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{z} - \boldsymbol{y} \rangle + \frac{L}{2}\|\boldsymbol{z} - \boldsymbol{y}\|^2 - f(\boldsymbol{x}) - \langle \nabla f(\boldsymbol{x}), \boldsymbol{z} - \boldsymbol{x} \rangle \\ &= f(\boldsymbol{y}) - f(\boldsymbol{x}) - \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \langle \nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x}), \boldsymbol{z} - \boldsymbol{y} \rangle + \frac{L}{2}\|\boldsymbol{z} - \boldsymbol{y}\|^2 \\ &= g_x(\boldsymbol{y}) + \langle \nabla g_x(\boldsymbol{y}), \boldsymbol{z} - \boldsymbol{y} \rangle + \frac{L}{2}\|\boldsymbol{z} - \boldsymbol{y}\|^2 \end{aligned}$$

注意到，$\nabla g_{\boldsymbol{x}}(\boldsymbol{y}) = \nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x})$ 对任意的 $\boldsymbol{y} \in \mathbb{R}^n$ 成立。如果 $\nabla g_{\hat{\boldsymbol{x}}}(\cdot) = 0$，结合函数 $g_x(\cdot)$ 的凸性，$\hat{\boldsymbol{x}}$ 是 $g_x(\cdot)$ 的全局最小解, 即

$$g_x(\hat{\boldsymbol{x}}) \leq g_x(\boldsymbol{z}), \quad \forall \boldsymbol{z} \in \mathbb{R}^n.$$

另 $\boldsymbol{y} \in \mathbb{R}^n$ 和 $\boldsymbol{v} \in \mathbb{R}^n$，且 $\|\boldsymbol{v}\| = 1$ 以及 $\langle \nabla g_x(\boldsymbol{y}), \boldsymbol{v} \rangle = \|\nabla g_x(\boldsymbol{y})\|$，将

$$\boldsymbol{z} = \boldsymbol{y} - \frac{\|\nabla g_x(\boldsymbol{y})\|}{L}\boldsymbol{v},$$

代入到

$$0 = g_x(\boldsymbol{x}) \leq g_x\left(\boldsymbol{y} - \frac{\|\nabla g_x(\boldsymbol{y})\|}{L}\boldsymbol{v}\right).$$

进一步根据函数 g_x 的性质，有

$$\begin{aligned} 0 &= g_{\boldsymbol{x}}(\boldsymbol{x}) \\ &\leq g_{\boldsymbol{x}}(\boldsymbol{y}) - \frac{\|\nabla g_{\boldsymbol{x}}(\boldsymbol{y})\|}{L}\langle \nabla g_x(\boldsymbol{y}), \boldsymbol{v} \rangle + \frac{1}{2L}\|\nabla g_{\boldsymbol{x}}(\boldsymbol{y})\|^2 \cdot \|\boldsymbol{v}\|^2 \\ &= g_{\boldsymbol{x}}(\boldsymbol{y}) - \frac{1}{2L}\|\nabla g_{\boldsymbol{x}}(\boldsymbol{y})\|^2 \\ &= f(\boldsymbol{y}) - f(\boldsymbol{x}) - \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle - \frac{2}{2L}\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|^2. \end{aligned}$$

这也就证明 iii) 成立。

iii)⇒iv)：分别对 $(\boldsymbol{x}, \boldsymbol{y})$ 和 $(\boldsymbol{y}, \boldsymbol{x})$ 应用 iii)，即

$$\begin{aligned} f(\boldsymbol{y}) &\geq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \tfrac{1}{2L}\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|^2 \\ f(\boldsymbol{x}) &\geq f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle + \tfrac{1}{2L}\|\nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x})\|^2 \end{aligned}$$

将上面两个不等式相加, 可以得到 iv)。
iv)⇒i)：针对 $\nabla f(\boldsymbol{x}) \neq \nabla f(\boldsymbol{y})$，根据 iv) 和柯西施瓦兹不等式，对任意的 $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$，

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \cdot \|\boldsymbol{x} - \boldsymbol{y}\| \geq \langle \nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle \geq \frac{1}{L} \|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|^2,$$

从而可以得到 $\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|$。
ii)⇒v)：另 $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ 且 $\lambda \in [0, 1]$。定义 $\boldsymbol{x}_\lambda = \lambda \boldsymbol{x} + (1 - \lambda)\boldsymbol{y}$，根据 ii) 可以得到

$$\begin{aligned} f(\boldsymbol{x}) &\leq f(\boldsymbol{x}_\lambda) + \langle \nabla f(\boldsymbol{x}_\lambda), \boldsymbol{x} - \boldsymbol{x}_\lambda \rangle + \tfrac{L}{2} \|\boldsymbol{x} - \boldsymbol{x}_\lambda\|^2, \\ f(\boldsymbol{y}) &\leq f(\boldsymbol{x}_\lambda) + \langle \nabla f(\boldsymbol{x}_\lambda), \boldsymbol{y} - \boldsymbol{x}_\lambda \rangle + \tfrac{L}{2} \|\boldsymbol{y} - \boldsymbol{x}_\lambda\|^2. \end{aligned}$$

将 x_λ 代入上面两个等式，可以得到

$$\begin{aligned} f(\boldsymbol{x}) &\leq f(\boldsymbol{x}_\lambda) + (1 - \lambda) \langle \nabla f(\boldsymbol{x}_\lambda), \boldsymbol{x} - \boldsymbol{y} \rangle + \tfrac{L(1-L)^2}{2} \|\boldsymbol{x} - \boldsymbol{y}\|^2, \\ f(\boldsymbol{y}) &\leq f(\boldsymbol{x}_\lambda) + \lambda \langle \nabla f(\boldsymbol{x}_\lambda), \boldsymbol{y} - \boldsymbol{x} \rangle + \tfrac{L\lambda^2}{2} \|\boldsymbol{x} - \boldsymbol{y}\|^2. \end{aligned}$$

上面第一个不等式左右两边乘以 λ 而第二个不等式左右两边乘以 $1 - \lambda$，进一步将其相加，可以得到 v) 。
v)⇒ii)：通过重新整写，v) 中的不等式等价于

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \frac{f(\boldsymbol{x} + (1 - \lambda)(\boldsymbol{y} - \boldsymbol{x})) - f(\boldsymbol{x})}{1 - \lambda} + \frac{L}{2} \lambda \|\boldsymbol{x} - \boldsymbol{y}\|^2.$$

令 $\lambda \to 1^-$，上式可以得到

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + f'(\boldsymbol{x}; \boldsymbol{y} - \boldsymbol{x}) + \frac{L}{2} \|\boldsymbol{x} - \boldsymbol{y}\|^2,$$

而 $f'(\boldsymbol{x}; \boldsymbol{y} - \boldsymbol{x}) = \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle$，从而可以得到 ii)。 □

2.6 无约束问题

考虑无约束最优化问题 $\min f(\boldsymbol{x})$, 假定目标函数 $f(\boldsymbol{x})$ 的一阶导数及二阶导数均存在, 定义 $f(\boldsymbol{x})$ 在点 $\boldsymbol{x}$ 处的梯度向量 $g(\boldsymbol{x})$ 与 Hesse 矩阵 G_x，记为:

$$g(\boldsymbol{x}) = \Delta f(\boldsymbol{x}), G(\boldsymbol{x}) = \Delta^2 f(\boldsymbol{x}) \tag{2.11}$$

无约束最优化问题的最优解分为全局最优解和局部最优解，下面给出定义:

定义 2.22 (全局最优解). 若对任意 $\boldsymbol{x} \in R, f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*)$，则称 $\boldsymbol{x}^*$ 为 $\min f(\boldsymbol{x})$ 的全局最优解。

定义 2.23 (局部最优解). 对 $\boldsymbol{x}^* \in \mathbb{R}^n$，若存在: $\varepsilon > 0$，使对任意 $\boldsymbol{x} \in \mathbb{R}^n$, 当 $\|\boldsymbol{x} - \boldsymbol{x}^*\| < \varepsilon$ 时，有 $f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*)$，则称 $\boldsymbol{x}*$ 为 $\min f(\boldsymbol{x})$ 的局部最优解。

定义 2.24 (稳定点). 满足 $g(\boldsymbol{x}) = 0$ 的点称为稳定点。极小点和极大点均为稳定点，既非极小点又非极大点的稳定点为鞍点。

若一个算法产生的点列 $\boldsymbol{x}_k$ 在某种范数意义下满足 $\lim\limits_{k \to +\infty} \|\boldsymbol{x}_k - \boldsymbol{x}^*\| = 0$，则称这个算法是收敛的。收敛算法的收敛速度有以下几种情形:

1. 线性收敛：若 $\lim\limits_{k \to +\infty} \dfrac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_k - \boldsymbol{x}^*\|} = \alpha$，当 $0 < \alpha < 1$，算法线性收敛。
2. 超线性收敛：上式中 $\alpha = 0$ 时，算法超线性收敛。
3. 二阶收敛：若 $\lim\limits_{k \to +\infty} \dfrac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_k - \boldsymbol{x}^*\|} = \alpha$，其中 α 为任意常数，则算法二阶收敛。

定义 2.25 (二次终止性). 当一个算法对任意正定二次函数，从任意初始点出发，可以经有限步迭代求得极小点，则称算法具有二次终止性。

2.6.1 线搜索方法

线搜索方法的思想是在 $\boldsymbol{x}_k$ 点求得下降方向 d_k，在沿 d_k 确定步长 α_k。其中下降方向 d_k, 为满足 $g_k^{\mathbb{T}} d < 0$ 的方向, 其算法如**??**:

算法 1 线搜索方法

输入: 初始点 $\boldsymbol{x}_0 \in \mathbb{R}^n, k = 0$
输出: 最优解 $\boldsymbol{x}_{k+1}$
while 未收敛 **do**
  确定 $f(\boldsymbol{x})$ 在 $\boldsymbol{x}_k$ 点的下降方向 d_k
  确定步长 α_k，使 $f(\boldsymbol{x}_k + \alpha_k d_k)$ 较之 $f(\boldsymbol{x}_k)$ 有某种意义的下降
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k d_k, k = k + 1$
end

2.6.2 线搜索准则

定理 2.26 (精确线搜索准则). 精确线搜索是指求步长 α，使得目标函数 $f()$ 在 d_k 方向上达到极小值

$$\min_{\alpha} f(\boldsymbol{x}_k + \alpha d_k) = \nabla f(\boldsymbol{x}_k + \alpha d_k)^{\mathbb{T}} d_k \Rightarrow g_{k+1}^{\mathbb{T}} d_k = 0 \tag{2.12}$$

定义 2.27 (Armijo-Goldstein 准则). [1980Curvilinear]

$$f(\boldsymbol{x}_k + \alpha d_k) \leq f(\boldsymbol{x}_k) + \rho g_k^T d_k \alpha, \rho \in (0, 1). \tag{2.13}$$

$$f(\boldsymbol{x}_k) + (1-\rho) g_k^T d_k \alpha \leq f(\boldsymbol{x}_k + \alpha d_k) \leq f(\boldsymbol{x}_k) + \rho g_k^T d_k \alpha, \rho \in (0, \frac{1}{2}). \tag{2.14}$$

选一点 $\boldsymbol{x}_k + 1 = \boldsymbol{x}_k - \alpha f'(\boldsymbol{x}_k)$, 使得目标函数逐渐下降。为了达到收敛到梯度为 0 点，需要满足：$f(\boldsymbol{x}_{k+1}) < f(\boldsymbol{x}_k)$，即函数值需呈下降趋势，$\boldsymbol{x}_{k+1}$ 不能距离 $\boldsymbol{x}_k$ 过于近，即步长 α 不能过小，同时确保目标函数值有足够下降

定义 2.28 (Wolfe 准则). Wolfe 准则主要用于线搜索，由 Armijo 准则和 curvature 条件共同组成。由上式**??**可知，Armijo 是充分下降条件，这个条件保证目标函数 $f(\boldsymbol{x}_k)$ 单调下降，如果目标函数有界，那么序列收敛，然而，只有 Armjio 条件无法保证迭代点序列 $\boldsymbol{x}_k$ 收敛以及局部最优条件。因为对于充分小的步长，Armijo 准则都是满足的

curvature 条件的作用就是拒绝掉满足 Armijo 准则的那些小步长，从而得到大步长，同时保证迭代点序列也是收敛的

$$f(\boldsymbol{x}_k + \alpha d_k) \leq f(\boldsymbol{x}_k) + \rho g_k^T d_k \alpha, \tag{2.15}$$

$$g(\boldsymbol{x}_k + \alpha d_k)^T d_k \geq \sigma g_k^T d_k. \tag{2.16}$$

其中 $0 < \rho < \sigma < 1$, 该准则**??**可以保证步长不过小。

定义 2.29 (强 Wolfe 准则).

$$f(\boldsymbol{x}_k + \alpha d_k) \leq f(\boldsymbol{x}_k) + \rho g_k^T d_k \alpha, |g(\boldsymbol{x}_k + \alpha d_k)^T d_k| \leq -\sigma g_k^T d_k. \tag{2.17}$$

其中 $0 < \rho < \sigma < 1$, 该准则的第二条可以保证步长不过小，强 Wolfe 准则可以使得 σ 取得越小，满足准则的 α 越接近精确线搜索的结果。

2.7 约束问题

约束最优化问题是指具有约束条件的非线性规划问题。求目标函数 $z = f(\boldsymbol{x}, \boldsymbol{y})$ 满足约束条件 $\varphi(\boldsymbol{x}, \boldsymbol{y}) = 0$ 的极值问题, 因此，约束最优化，也称条件极值。约束最优化问题，可采用消元法、拉格朗日乘子法或罚函数法，将其化为无约束最优化问题求解。

2.8 凸优化问题

首先考虑一般标准型最优化问题:

$$\begin{aligned} \min_x \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & f_i(\boldsymbol{x}) \leq 0, i = 1, \cdots, m \\ & h_j(\boldsymbol{x}) = 0, j = 1, \cdots, p \end{aligned} \tag{2.18}$$

其中 $\boldsymbol{x} \in \mathbb{R}^n$ 称为该最优化问题的变量；函数 $f : \mathbb{R}^n \to \mathbb{R}$ 称为该最优化问题的目标函数或代价函数；函数 $f_i : \mathbb{R}^n \to \mathbb{R}$ 称为该最优化问题的不等式约束函数；函数 $h_j : \mathbb{R}^n \to \mathbb{R}$ 称为该最优化问题的等式约束函数。一般地, 我们记该最优化问题 **(??)** 的最优目标函数值为 p^*, 即

$$p^* = \min \left\{ f(\boldsymbol{x}) \mid f_i(\boldsymbol{x}) \leq 0, i = 1, \cdots, m, h_j(\boldsymbol{x}) = 0, j = 1, \cdots, p \right\}.$$

如果 $p^* = \infty$，那么称该优化问题是不可行的 (满足约束条件的可行点集合为空集)；如果 $p^* = -\infty$，那么称该最优化问题是无下界的[2004Convex]。进一步，若 $\boldsymbol{x} \in \text{dom}(f)$ 且满足所有不等式与等式约束，则认为 $\boldsymbol{x}$ 是一个可行解 (feasible solution)，所有可行解构成的集合为 $\boldsymbol{X}_{fea}$。如果 $f(\boldsymbol{x}) = p^*$，则该可行解 $\boldsymbol{x}$ 为最优解，而最优解集合可以记为 $\boldsymbol{X}_{\text{opt}}$。如果存在 $r > 0$，且 $\hat{\boldsymbol{x}}$ 是下面问题的最优解:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & f_i(\boldsymbol{x}) \leq 0, i = 1, \cdots, m, \\ & h_j(\boldsymbol{x}) = 0, j = 1, \cdots, p, \\ & \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2 \leq r, \end{aligned}$$

那么这个解 $\hat{x}$ 为原始最优化问题 **(??)** 的局部最优解。特别地，如果只考虑最优化问题 **(??)** 的可行解集是否为空，可以有针对性地定义可行性问题，即

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

该可行性问题等价于

$$\begin{aligned} \min \quad & 0 \\ \text{s.t.} \quad & f_i(\boldsymbol{x}) \leq 0, i = 1, \cdots, m \\ & h_j(\boldsymbol{x}) = 0, j = 1, \cdots, p_\circ \end{aligned} \tag{2.19}$$

如果最优化问题 **(??)** 的最优目标函数 p^* 为 0，则该问题的所有约束是可行的，并且所有的可行解 $\boldsymbol{x} \in \mathcal{X}_{fea}$ 对于问题 **(??)** 都是最优的；但是如果 p^* 为 ∞，则该问题的约束是不可行的。

考虑下面特别的最优化问题:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & f_i(\boldsymbol{x}) \leq 0, i = 1, \cdots, m, \\ & \boldsymbol{a}_j^\top \boldsymbol{x} = b_j, j = 1, \cdots, p, \end{aligned} \tag{2.20}$$

其中函数 $f, f_1, \cdots, f_m$ 都是凸函数，而且所有的等式约束是线性约束。该最优化问题 **(??)** 被认为是凸优化问题。该问题 **(??)** 可以通过下面的更紧凑的形式表示:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & f_i(\boldsymbol{x}) \leq 0, i = 1, \cdots, m, \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \end{aligned} \tag{2.21}$$

例 2.30. 凸优化问题的相关例子:

(1) 线性规划 (Linear programming, LP)[1963Linear]

$$\begin{aligned} \min \quad & \boldsymbol{c}^\top \boldsymbol{x} + d \\ \text{s.t.} \quad & \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h}, \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned}$$

(2) 二次规划问题 (Quadratic programming, QP)[1995Sequential]

$$\begin{aligned} \min \quad & \tfrac{1}{2}\boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^\top \boldsymbol{x} + r \\ \text{s.t.} \quad & \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h}, \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned}$$

(3) 带二次约束二次规划问题 (Quadratically constrained Quadratic programming, QCQP)[1982Quadratically]

$$\begin{aligned} \min \quad & \tfrac{1}{2}\boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^\top \boldsymbol{x} + r \\ \text{s.t.} \quad & \tfrac{1}{2}\boldsymbol{x}^\top \boldsymbol{P}_i\boldsymbol{x} + \boldsymbol{q}_i^\top \boldsymbol{x} + r_i \le 0, i = 1, \cdots, m \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned}$$

(4) 最小二乘问题 (Least square problem)[1989Least]

$$\min \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2.$$

2.9 Weierstrass 定理

考虑优化问题

$$\begin{aligned} & \min_{x\in\mathbb{R}^n} f(x), \\ & \text{s.t.} \qquad x \in \mathcal{X} \end{aligned} \tag{2.22}$$

定理 2.31 (Weierstrass 定理). 考虑适当闭函数 $f : \mathcal{X} \to (-\infty, +\infty]$，假设下面 3 个条件中的任意一个成立:

(1) $\mathrm{dom}(f) := \{x \in \mathcal{X} : f(x) < +\infty\}$

(2) 存在常数 γ 使集合 $C_\gamma := \{x \in \mathcal{X} : f(x) \le \gamma\}$ 是非空有界的

(3) $f(\cdot)$ 满足对任意的点列 $\{x^k\} \subset \mathcal{X}, \|x^k\| \to +\infty$，都有 $\lim\limits_{k\to\infty} f(x^k) = +\infty$

则对于上述的优化问题存在非空的最小点集 $\{x \in \mathcal{X} | f(x) \le f(y), \forall y \in \mathcal{X}\}$

证明: 假设条件 (2) 成立，假设 $t = -\infty$，

则存在点列 $\{x^k\} \subset C_\gamma$，使 $\lim\limits_{k\to\infty} f(x^k) = t = -\infty$.

也就是说，点列 $\{x^k\}$ 一定存在聚点 x^*，即在 x^* 的任意邻域内，都有无穷多个点属于集合 C_γ.

那么，$f(x^*) \le t = -\infty$，与适当函数的定义矛盾，因此假设不成立，即 $t > -\infty$.

假设条件 (1) 成立，则 $\mathrm{dom}(f)$ 有界，

由于 $f(\cdot)$ 是适当函数，因此存在 $x_0 \in \mathcal{X}$ 使得 $f(x_0) < +\infty$，

令 $\gamma = f(x_0)$，那么集合 C_γ 为非空有界的

假设条件 (3) 成立，则存在 $x_0 \in \mathcal{X}$ 使得 $f(x_0) < +\infty$ 与 $\gamma = f(x_0)$，

假设 C_γ 无界，那么存在点列 $\{x^k\} \subset C_\gamma$ 满足 $\lim\limits_{k\to\infty} \|x^k\| = +\infty$，

于是由条件 (3) 可知，$\lim\limits_{k\to\infty} f(x^k) = +\infty$，与 $f(x) \le \gamma$ 矛盾，于是假设不成立，即得到条件 (2). □

定理 2.32. 凸优化问题 **(??)** 的任何局部最优解也是全局最优解。

证明: 如果对于可行解 $\boldsymbol{x} \in \boldsymbol{X}_{fea}$ 是凸优化问题 **(??)** 的一个局部最优解但不是全局最优解，即存在一个可行解 $\boldsymbol{y} \in \boldsymbol{X}_{fea}$ 满足

$$f(\boldsymbol{y}) < f(\boldsymbol{x}).$$

因为 $\boldsymbol{x}$ 是局部最优化，所以存在一个 $r > 0$ 使得

$$\forall \boldsymbol{z} \in \{\boldsymbol{z} \mid \boldsymbol{z} \in \boldsymbol{X}_{\text{fea}}, \|\boldsymbol{z} - \boldsymbol{x}\|_2 \le r\} \quad \Rightarrow \quad f(\boldsymbol{z}) \ge f(\boldsymbol{x}),$$

进一步考虑

$$\hat{\boldsymbol{z}} = \theta \boldsymbol{y} + (1-\theta)\boldsymbol{x}, \quad \theta = \frac{r}{2\|\boldsymbol{y} - \boldsymbol{x}\|_2}$$

我们得到

-$\|\boldsymbol{y} - \boldsymbol{x}\| > r$，所以 $0 < \theta < \frac{1}{2}$;

-$\hat{\boldsymbol{z}}$ 是两个可行点 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 的凸组合, 所以 $\hat{\boldsymbol{z}} \in \boldsymbol{X}_{fea}$;

- 因为 $\|\hat{\boldsymbol{z}} - \boldsymbol{x}\|_2 = \frac{r}{2} < r$，所以

$$f(\hat{\boldsymbol{z}}) \le \theta f(\boldsymbol{y}) + (1-\theta) f(\boldsymbol{x}) < f(\boldsymbol{x}),$$

这与 $\boldsymbol{x}$ 是局部最优的假设是矛盾的，所以原定理的结论成立。

3. 梯度下降法

梯度下降法 (Gradient Descent, GD) 是求解无约束优化问题的一种最常见的方法[6302929]，在最优化、统计学以及机器学习等领域都有着广泛的应用。对于无约束最优化问题:

$$\min_{x} \{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathbb{R}^n\}, \tag{3.1}$$

其中目标函数 $f : \mathbb{R}^n \to \mathbb{R}$ 是连续可微 $L-$ 光滑的函数。最优化问题 **(??)** 可以看作为一般问题 **(??)** 不加约束的特殊情况。由推论 **(??)** 可知，凸优化问题的最优解等价于梯度为零的解，即 $\boldsymbol{x}$ 满足 $\nabla f(\boldsymbol{x}) = 0$。如果 $\nabla f(\boldsymbol{x}) \neq 0$，那么根据函数 $\boldsymbol{f(x)}$ 在 $\boldsymbol{x}$ 处泰勒展开的性质可知，一定存在 δ，使得

$$f(\boldsymbol{x} - \alpha \nabla f(\boldsymbol{x})) < f(\boldsymbol{x}), \quad \forall \alpha \in (0, \delta) \tag{3.2}$$

(??) 也表示 $\boldsymbol{x}$ 梯度的反方向，即 $-\nabla f(\boldsymbol{x})$，是目标函数值的下降方向。进一步，可以将这一下降现象推广到更一般的情形。如果方向 $\boldsymbol{d} \in \mathbb{R}^n$ 满足

$$\nabla f(\boldsymbol{x})^\top \boldsymbol{d} < 0,$$

即方向 d 与梯度方向 $\nabla f(\boldsymbol{x})$ 的夹角大于 90°，此时一定存在 σ，满足:

$$f(\boldsymbol{x} + \alpha \boldsymbol{d}) < f(\boldsymbol{x}), \quad \forall \alpha \in (0, \delta) \tag{3.3}$$

沿着满足该性质的方向 d 更新 x 可以使目标函数下降，从而达到减小函数值的目的。

3.1 梯度下降法

一种凸优化模型通常通过人工设计迭代格式的方式来构建优化变量迭代序列 $\left\{\boldsymbol{x}^k\right\}$，从而寻找问题的最优解，理论上需要建立所得到的迭代序列收敛到最优解的证明[6302929]。梯度下降法，作为最简单常用的梯度类方法，是典型的以迭代格式进行最优化问题求解的算法，其具体的第 $k+1$ 步的迭代格式如下:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha_k \nabla f\left(\boldsymbol{x}^k\right), \tag{3.4}$$

其中 k 对应于迭代步数，$\alpha_k > 0$ 称为步长 (step-size) 或学习率 (learning rate)。梯度下降法的迭代格式 **(??)** 也可以从最优性条件的角度来理解，无约束优化问题 **(??)** 的最优性条件等

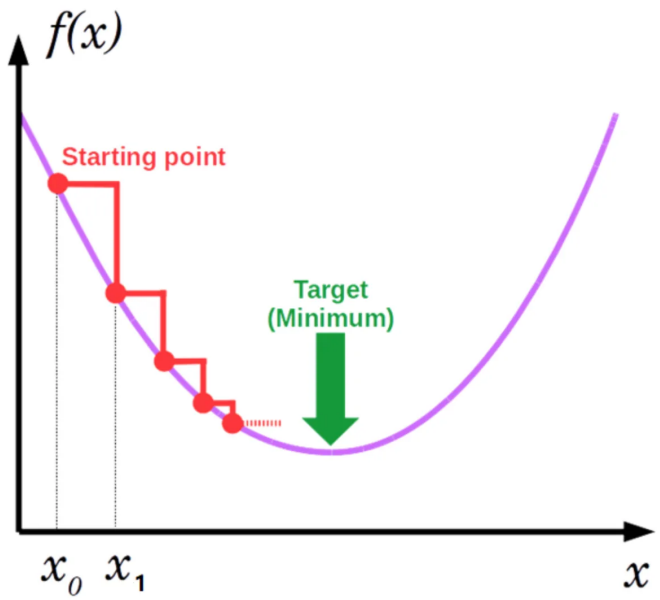

图 3-1: 梯度下降法实例

价于

$$\nabla f\left(\boldsymbol{x}^*\right)=0 \quad \Leftrightarrow \quad \boldsymbol{x}^*=\boldsymbol{x}^*-\alpha \nabla f\left(\boldsymbol{x}^*\right),$$

以不动点迭代的形式可以将该等价条件改写为梯度下降算法的迭代格式。如果所得到的序列 $\left\{\boldsymbol{x}^k\right\}$ 是收敛的且收敛到 $\boldsymbol{x}^*$，步长 α_k 也收敛到 α_∞，那么对迭代格式 **(??)** 中取 $k \to \infty$ 可以得到

$$\boldsymbol{x}^*=\boldsymbol{x}^*-\alpha_\infty \nabla f\left(\boldsymbol{x}^*\right).$$

该式子等价于 $\nabla f\left(x^*\right)=0$，这也说明 x^* 满足一阶最优性条件，即成为最优化问题 **(??)** 的最优解。梯度下降法是针对无约束最优化问题求解的最基本算法，除了迭代格式外，步长 k 的设计方式影响问题的求解效率和能力。下面介绍一系列常用步长设计方式[2009Accelerated]：

(1) 常数步长准则：$\alpha_k=\ell$；

(2) 极小化准则：$\alpha_k=\arg\min_{\alpha \geq 0} f\left(\boldsymbol{x}^k-\alpha \nabla f\left(\boldsymbol{x}^k\right)\right), \alpha \in(0, \delta)$；

(3) 逐步减少 (diminishing) 步长准则：步长 α_k 满足

$$\lim_{k \to \infty} \alpha_k=0, \quad \sum_{k=0}^{\infty} \alpha_k=\infty;$$

(4) Armijo[1999Convergence] 准则：$\delta \in\left(0, \frac{1}{2}\right)$，令 α 从初始值 s 开始以 $\beta \in(0,1)$ 的倍数逐步减小，即

$$\left\{s, \beta s, \beta^2 s, \cdots\right\},$$

直到 $\beta^m s$ 且首次满足

$$f\left(x^k\right)-f\left(x^k-\alpha\nabla f\left(x^k\right)\right)\geq-\delta\alpha\nabla f\left(x^k\right)^T\nabla f\left(x^k\right).$$

Armijo 准则是一种常用的步长设计准则[1999Convergence]，其通过定量标准来协助选择合适步长，通过有限步最终可以找到满足条件的步长，常用于非凸优化问题的求解计算。

3.2 梯度下降法收敛性分析

对于最优化方法的收敛性理论分析，主要包括算法收敛性、算法局部收敛速度以及算法全局迭代复杂度等方面[2002ConvergenceAnalysis]。算法收敛性 (convergence) 指的是算法得到的迭代序列 $\left\{x^k\right\}$ 的序列收敛性，如是否有聚点、聚点是否满足最优性条件以及序列是否全局收敛等。算法局部收敛速度 (convergence rate) 指的是在已证明算法迭代序列收敛的前提下，算法迭代序列可以以某种可量化的速度收敛到最优解 (可能是局部最优解)，如线性收敛速度、超线性收敛速度、二次收敛速度等。收敛速度通常考虑的只是局部性质，即算法迭代到一定步数后序列所体现出来的性质。与此同时，另一种算法全局迭代复杂度 (iteration complexity) 刻画的是算法从初始点开始经过一定迭代步数后所能到达的“最优化”程度 (距离最优解或最优函数值的程度)，即经过 k 步迭代后，算法得到的序列 $\left\{\boldsymbol{x}^k\right\}$ 满足性质：

$$f\left(\boldsymbol{x}^k\right)-f\left(\boldsymbol{x}^*\right)\leq\varepsilon. \tag{3.5}$$

对于梯度下降法 **(??)**，在最优化问题 **(??)** 的目标函数 $f(\cdot)$ 是连续可微 $L-$ 光滑的基础上，假设 $f(\cdot)$ 为凸函数。其中 $f(\cdot)$ 的 $L-$ 光滑性质，意味着

$$\|\nabla f(\boldsymbol{x})-\nabla f(\boldsymbol{y})\|\leq L\|\boldsymbol{x}-\boldsymbol{y}\|,\quad\forall\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^n.$$

定理 3.1 (收敛性). 对于无约束最优化问题 **(??)** 的目标函数 $f(\cdot)$ 是连续可微 $L-$ 光滑的凸函数，序列 $\left\{\boldsymbol{x}^k\right\}$ 为梯度下降法 **(??)** 得到的迭代序列，且步长 $\alpha_k\in\left(0,\frac{2}{L}\right)$，那么该序列 $\left\{\boldsymbol{x}^k\right\}$ 收敛到 $\boldsymbol{x}^*$，其中 $\boldsymbol{x}^*$ 是最优化问题 **(??)** 的全局最优解。梯度下降法的迭代复杂度为

$$f\left(\boldsymbol{x}^k\right)-f\left(\boldsymbol{x}^*\right)\leq\mathcal{O}\left(\frac{1}{k}\right).$$

证明: 首先，根据函数 $f(\cdot)$ 的 $L-$ 光滑性质以及下降引理 **(??)**，对于任意的 $\boldsymbol{x},\boldsymbol{y}\in\mathcal{D}$ 有

$$f(\boldsymbol{y})\leq f(\boldsymbol{x})+\langle\nabla f(\boldsymbol{x}),\boldsymbol{y}-\boldsymbol{x}\rangle+\frac{L}{2}\|\boldsymbol{x}-\boldsymbol{y}\|^2.$$

令 $\boldsymbol{x}=\boldsymbol{x}^k$ 、$\boldsymbol{y}=\boldsymbol{x}^{k+1}$，可以得到

$$f\left(\boldsymbol{x}^{k+1}\right) \leq f\left(\boldsymbol{x}^k\right)+\nabla f\left(\boldsymbol{x}^k\right)^{\top}\left(\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\right)+\frac{L}{2}\left\|\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\right\|^2 .$$

进一步将算法迭代格式 $\boldsymbol{x}^{k+1}=\boldsymbol{x}^k-\alpha_k \nabla f\left(\boldsymbol{x}^k\right)$ 代入上式，可以得到

$$f\left(\boldsymbol{x}^{k+1}\right) \leq f\left(\boldsymbol{x}^k\right)-\left(\alpha_k-\frac{\alpha_k^2 L}{2}\right)\left\|\nabla f\left(\boldsymbol{x}^k\right)\right\|^2 . \tag{3.6}$$

因为 $\alpha_k \in\left(0, \frac{2}{L}\right)$ ，所以

$$\alpha_k-\frac{\alpha_k^2 L}{2}>0, \tag{3.7}$$

可以发现算法从 $\boldsymbol{x}^k$ 迭代到 $\boldsymbol{x}^{k+1}$，所对应的目标函数值是下降的，且最坏情况下降

$$\left(\alpha_k-\frac{\alpha_k^2 L}{2}\right)\left\|\nabla f\left(\boldsymbol{x}^k\right)\right\|^2 .$$

所以式 **(??)** 也被称为充分下降性质 (sufficient decrease)[2009Accelerated]。再次利用算法的迭代格式 $\boldsymbol{x}^{k+1}=\boldsymbol{x}^k-\alpha_k \nabla f\left(\boldsymbol{x}^k\right)$，并结合第 2 章提到的凸函数的性质，可以得到

$$\begin{aligned}
\left\|\boldsymbol{x}^{k+1}-\boldsymbol{x}^*\right\|^2 &= \left\|\boldsymbol{x}^k-\alpha_k \nabla f\left(\boldsymbol{x}^k\right)-\boldsymbol{x}^*\right\|^2 \\
&= \left\|\boldsymbol{x}^k-\boldsymbol{x}^*\right\|^2-\alpha_k \nabla f\left(\boldsymbol{x}^k\right)^{\top}\left(\boldsymbol{x}^k-\boldsymbol{x}^*\right)+\alpha_k^2\left\|\nabla f\left(\boldsymbol{x}^k\right)\right\|^2 \\
&\leq \left\|\boldsymbol{x}^k-\boldsymbol{x}^*\right\|^2+\alpha_k\left[f\left(\boldsymbol{x}^*\right)-f\left(\boldsymbol{x}^k\right)\right]+\alpha_k^2\left\|\nabla f\left(\boldsymbol{x}^k\right)\right\|^2 .
\end{aligned} \tag{3.8}$$

结合不等式 **(??)** 和不等式 **(??)**，可以进一步得到

$$k\left[f\left(\boldsymbol{x}^k\right)-f\left(\boldsymbol{x}^*\right)\right] \leq \quad \sum_{i=1}^k\left[f\left(\boldsymbol{x}^i\right)-f\left(\boldsymbol{x}^*\right)\right],$$

由 $\alpha_k \leq \frac{2}{L}$ 和不等式 **(??)** 可得：

$$\begin{aligned}
k\left[f\left(\boldsymbol{x}^k\right)-f\left(\boldsymbol{x}^*\right)\right] \leq & \quad \sum_{i=1}^k\left[\frac{L}{2}\left(\left\|\boldsymbol{x}^i-\boldsymbol{x}^*\right\|^2-\left\|\boldsymbol{x}^{i+1}-\boldsymbol{x}^*\right\|^2\right)+\alpha_i^2\left\|\nabla f\left(\boldsymbol{x}^i\right)\right\|^2\right] \\
= & \quad \frac{L}{2}\left(\left\|\boldsymbol{x}^1-\boldsymbol{x}^*\right\|^2-\left\|\boldsymbol{x}^{k+1}-\boldsymbol{x}^*\right\|^2\right)+\sum_{i=1}^k \alpha_i^2\left\|\nabla f\left(\boldsymbol{x}^i\right)\right\|^2,
\end{aligned}$$

再由式 **(??)** 和式 **(??)** 可得：

$$\begin{aligned} k\left[f\left(\boldsymbol{x}^{k}\right)-f\left(\boldsymbol{x}^{*}\right)\right] &\leq \frac{L}{2}\left(\left\|x^{1}-x^{*}\right\|^{2}-\left\|x^{k+1}-x^{*}\right\|^{2}\right)+\sum_{i=1}^{k}\frac{8}{L}\left[f\left(\boldsymbol{x}^{i}\right)-f\left(\boldsymbol{x}^{i+1}\right)\right] \\ &\leq \quad \frac{L}{2}\left\|\boldsymbol{x}^{1}-\boldsymbol{x}^{*}\right\|^{2}+\frac{8}{L}f\left(\boldsymbol{x}^{1}\right). \end{aligned} \tag{3.9}$$

其中该不等式的最右边为一个常数。

根据式 **(??)**，我们有

$$\frac{1}{2L}\sum_{i=1}^{k}\left\|\nabla f\left(\boldsymbol{x}^{i}\right)\right\|^{2}=\sum_{i=1}^{k}\left(\alpha_{i}-\frac{\alpha_{i}^{2}L}{2}\right)\left\|\nabla f\left(\boldsymbol{x}^{i}\right)\right\|^{2}\leq\sum_{i=1}^{k}\left[f\left(\boldsymbol{x}^{i}\right)-f\left(\boldsymbol{x}^{i+1}\right)\right]\leq f\left(\boldsymbol{x}^{1}\right),$$

观察右边的不等式，由于式 **(??)**，故 $\sum_{i=1}^{k}\left(\alpha_{i}-\frac{\alpha_{i}^{2}L}{2}\right)$ 随着 k 的增大而增大，但它小于 $f\left(\boldsymbol{x}^{1}\right)$ 永远成立，因此可以得到

$$\nabla f\left(\boldsymbol{x}^{k}\right)\overset{k\to\infty}{\longrightarrow}\mathbf{0}.$$

根据式 **(??)** 可知，

$$\left\|\boldsymbol{x}^{k+1}-\boldsymbol{x}^{*}\right\|^{2}\leq\left\|\boldsymbol{x}^{1}-\boldsymbol{x}^{*}\right\|^{2}+\sum_{i=1}^{k}\alpha_{i}^{2}\left\|\nabla f\left(\boldsymbol{x}^{i}\right)\right\|^{2}<\infty,$$

因此得到序列 $\left\{\boldsymbol{x}^{k}\right\}$ 是有界的，那么其一定有收敛子列 $\left\{\boldsymbol{x}^{k(\ell)}\right\}_{\ell=1}^{\infty}$，$\ell$ 为迭代步数，聚点记作 $\hat{\boldsymbol{x}}^{*}$，即

$$\boldsymbol{x}^{k(\ell)}\overset{\ell\to\infty}{\longrightarrow}\hat{\boldsymbol{x}}^{*},$$

那么 $\hat{\boldsymbol{x}}^{*}$ 满足 $\nabla f\left(\hat{\boldsymbol{x}}^{*}\right)=\mathbf{0}$。由于

$$\left\|\boldsymbol{x}^{k(\ell)+1}-\hat{\boldsymbol{x}}^{*}\right\|\leq\left\|\boldsymbol{x}^{k(\ell)}-\hat{\boldsymbol{x}}^{*}\right\|+\left\|\boldsymbol{x}^{k(\ell)+1}-\boldsymbol{x}^{k(\ell)}\right\|,$$

那么 $x^{k(\ell)+1}$ 也收敛到 $\hat{x}^{*}$，所以我们有

$$\boldsymbol{x}^{k}\overset{k\to\infty}{\longrightarrow}\hat{\boldsymbol{x}}^{*}$$

进一步根据式 (??) 可知，该序列 $\left\{x^k\right\}$ 的全局迭代复杂度[6302929] 为

$$f\left(\boldsymbol{x}^k\right)-f\left(\hat{\boldsymbol{x}}^*\right) \leq \mathcal{O}\left(\frac{1}{k}\right),$$

即为 $\left(\frac{1}{k}\right)$-迭代复杂度。 □

3.3 梯度下降法的求解应用

例 3.2 (实践题). 考虑最小二乘问题，即

$$\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|^2,$$

其中 $\boldsymbol{A} \in \mathbb{R}^{1000\times 1000}, \boldsymbol{x} \in \mathbb{R}^{1000}$。实现求解该最小二乘问题的梯度下降法，其中采用常数步长。请输出所得到的近似解 $\boldsymbol{x}^k$ 以及算法收敛曲线。

解: 根据题意，$\boldsymbol{b} \in \mathbb{R}^{1000}$。
首先，随机生成对应维数的矩阵 $\boldsymbol{A}$ 和最优解 $\boldsymbol{x}^*$，并进一步根据 $\boldsymbol{b}=\boldsymbol{A}\boldsymbol{x}^*$ 得到 $\boldsymbol{b}$ 。
算法的迭代格式如下:

$$\boldsymbol{x}^{k+1}=\boldsymbol{x}^k-\alpha_k \nabla f\left(\boldsymbol{x}^k\right),$$

其中，$\nabla f\left(\boldsymbol{x}^k\right)=2\times A^T\times(Ax-b)$，算法终止条件设置为

$$\frac{\left\|\boldsymbol{x}^k-\boldsymbol{x}^*\right\|_2}{\left\|\boldsymbol{x}^*\right\|_2}<10^{-4},$$

若 numpy 随机种子设置为 1234，则收敛曲线如图所示，其中横坐标表示迭代步数，纵坐标为 $\log\left(\left\|\boldsymbol{x}^k-\boldsymbol{x}^*\right\|_2\right)$。

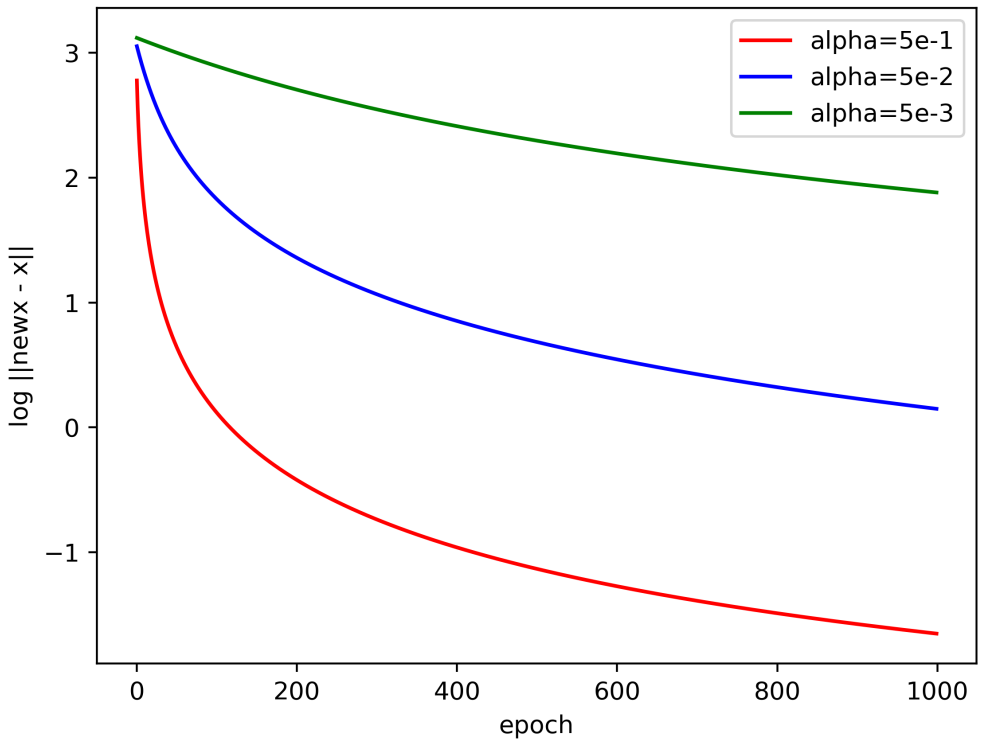

图 3-2: Python 实现梯度下降法求解最小二乘问题

4. 随机梯度类方法

随机梯度类方法是处理大规模最优化问题的有效方法，最早在 1951 年由 Robbins 和 Monro 所提出[SGD]，多用于支持向量机、逻辑回归（LR）等凸损失函数下的线性分类器的学习，并且随机梯度类方法已成功应用于文本分类和自然语言处理中经常遇到的大规模和稀疏机器学习问题。以大规模监督机器学习问题为例，监督学习任务所对应的训练数据集包含 n 条数据 $\{(\boldsymbol{x}_i, y_i), i = 1, \cdots, n\}$。假设预测模型为 $h(\boldsymbol{x}, \mathrm{w})$，其中参数 w 为预测模型的参数，且 $\mathrm{w} \in \mathbb{R}^d$。带正则项的监督学习任务可以建模为以下形式，其中 $\ell(x)$ 为凸函数，

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \ell\left(h\left(\boldsymbol{x}_i, \mathrm{w}\right); y_i\right) + \lambda h(\mathrm{w}). \tag{4.1}$$

随机梯度类方法尤其适用于数据量 n 是大规模的情形。对于上面介绍的监督学习任务 (**??**)，其参数化模型可以被归纳为

$$\min_{\mathrm{x}\in\mathbb{R}^d} \underbrace{\frac{1}{n}\sum_{i=1}^{n} f_i(\mathrm{x})}. \tag{4.2}$$

该问题通常被称为有限求和经验风险极小化问题，其也是机器学习任务中经常面对的结构型最优化问题，但是机器学习任务更重要的核心目标是极小化测试任务中的风险（损失）。在解决机器学习任务时，没有必要过度强调对训练阶段任务的求解计算精度，而是需要更关注测试阶段的任务，即所得到的“最优解”在测试阶段的表现。因此，针对以监督学习为代表的机器学习任务，所设计的算法应该具备快速高效求解能力，适合处理海量数据，而无需为过高的求解精度而付出较多的计算迭代。

4.1 经典随机梯度法

针对结构型最优化问题 (**??**)，即

$$\min_{\mathrm{x}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} f_i(\mathrm{x}).$$

针对该问题的经典随机梯度法 (Stochastic gradient descent method, SGD)[SGD] 的基本迭代框架为

$$\mathrm{x}^{k+1} = \mathrm{x}^k - \alpha_k \nabla f_{i(k)}\left(\mathrm{x}^k\right), \tag{4.3}$$

其中 $i(k)$ 是从 $\{1,2,\cdots,n\}$ 中允许放回的按均匀分布 (每个 f_i 被选择概率是 $\frac{1}{n}$) 随机选取的。定义 Polyak-Ruppert 平均[2021Statistical] 为

$$\bar{\theta}_k = \frac{1}{k+1}\sum_{j=1}^{k}\theta^j. \tag{4.4}$$

对于该基本随机梯度法可以证明其对于凸优化问题的收敛性和收敛速度。如果函数 f_i 均为凸函数且为 $L-$ 光滑的，且令 $g(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathrm{x})$，那么可以得到期望的收敛速度:

$$\mathbb{E}\left[g\left(\overline{\mathrm{x}}_k\right) - g\left(\mathrm{x}^*\right)\right] \leq \mathcal{O}\left(\frac{1}{\sqrt{k}}\right), \quad \alpha_k = \frac{1}{L\sqrt{k}}. \tag{4.5}$$

该结论同时说明了算法的收敛性和收敛速度[SGD]，可以看出对于一般的凸优化问题，基本随机梯度法比经典梯度下降法的收敛速度慢，但随机梯度法每个迭代步不需要计算整体目标函数 $g(\mathrm{w})$ 的梯度，而是从样本中随机抽出一组，训练后按梯度更新一次，然后反复抽取和更新，在样本量及其大的情况下，无需训练完所有的样本就可以获得一个损失值在可接受范围之内的模型了。进一步，如果函数 $g(\mathrm{w})$ 为 $\mu-$ 强凸函数，那么可以得到

$$\mathbb{E}\left[g\left(\overline{\mathrm{x}}_k\right) - g\left(\mathrm{x}^*\right)\right] \leq \mathcal{O}\left(\frac{1}{\mu k}\right), \quad \alpha_k = \frac{1}{\mu k}, \tag{4.6}$$

其可以达到 $\mathcal{O}(1/k)$ 迭代复杂，但是相比于梯度下降法在强凸假设前提下的线性收敛速度仍要慢。

4.2 随机平均梯度法

从基本随机梯度法来看，每一个迭代步都只利用当前迭代步所选择到的数据进行更新，而未充分利用历史梯度信息。随机平均梯度法的基本思想是将计算过的 f_i 的梯度都保存在内存中，并不断更新且用于算法设计，其可以看作是随机版本的增量平均梯度方法。随机平均梯度法（Stochased average gradient method, SAG)[SAG] 的具体格式如算法 **(??)** 所示。

随机平均梯度法与经典随机梯度法一致，在每个迭代步只需计算一次单个函数梯度，但是随机平均梯度法需要在内存中存储 n 个 d 维梯度向量，在存储需求方面比随机梯度法显著增加。理论上，随机平均梯度法在 $\mu-$ 强凸假设下可以达到线性收敛速度，即

$$\mathbb{E}\left[g\left(\overline{\mathrm{x}}_k\right) - g\left(\mathrm{x}^*\right)\right] \leq \mathcal{O}\left\{\left(1 - \min\left\{\frac{1}{8n}, \frac{\mu}{16L}\right\}\right)^k\right\},$$

算法 2 随机平均梯度法

输入: 目标函数 f_i

输出: 最优解 $\tilde{\mathrm{x}}$

for $i = 1, 2, \cdots$ **do**

函数 f_i 在第 k 步迭代所对应的梯度记作 $\boldsymbol{y}_i^k$

随机可放回的选择 $i(k) \in \{1, \cdots, n\}$

更新梯度信息，即

$$\begin{cases} \boldsymbol{y}_j^{k+1} = \nabla f_j\left(\mathrm{x}^k\right), & j = i(k) \\ \boldsymbol{y}_j^{k+1} = \boldsymbol{y}_j^k, & j \neq i(k) \end{cases}$$

更新参数变量 x^{k+1}，即

$$\mathrm{x}^{k+1} = \mathrm{x}^k - \frac{\alpha_k}{n} \sum_{j=1}^{n} y_j^{k+1},$$

end

而这一优势也是因为对已计算历史梯度信息的充分利用所获得的[SAG]。

4.3 方差减小随机梯度法

方差减小技巧是统计学中的一种常用技巧，其核心是通过设计一个新的随机变量，并结合采样技术降低已知随机变量的方差[SVRG]。假设 $\mathbb{X}$ 为已知随机变量，给定一个新的随机变量 $\mathbb{Y}$ 以及它的期望为 $\mathbb{E}[\mathbb{Y}]$，定义新的随机变量 $\mathbb{Z}_\alpha$ 使得

$$\mathbb{Z}_\alpha = \alpha(\mathbb{X} - \mathbb{Y}) + \mathbb{E}[\mathbb{Y}],$$

该随机变量 $\mathbb{Z}_\alpha$ 的期望是

$$\mathbb{E}\left[\mathbb{Z}_\alpha\right] = \alpha\mathbb{E}[\mathbb{X}] + (1-\alpha)\mathbb{E}[\mathbb{Y}],$$

而方差是

$$\mathrm{Var}\left[\mathbb{Z}_\alpha\right] = \alpha^2[\mathrm{Var}[\mathbb{X}] + \mathrm{Var}[\mathbb{Y}] - 2\,\mathrm{Cov}(\mathbb{X}, \mathbb{Y})].$$

值得注意的是如果 $\alpha = 1$，那么可以得到 $\mathbb{E}\left[\mathbb{Z}_\alpha\right] = \mathbb{E}[\mathbb{X}]$，即随机变量 $\mathbb{Z}_\alpha$ 可以看作为随机变量 $\mathbb{X}$ 的一种无偏随机变量估计。当 $\mathbb{Y}$ 与 $\mathbb{X}$ 负相关时，此时可以得到 $\mathrm{Var}\left[\mathbb{Z}_\alpha\right] < \mathrm{Var}[\mathbb{X}]$。如果 $\alpha < 1$，那么虽然 $\mathbb{Z}_\alpha$ 是有偏的，但是 $\mathrm{Var}\left[\mathbb{Z}_\alpha\right]$ 依然比 $\mathrm{Var}[\mathbb{X}]$ 小。

为什么以及如何利用方差减小技巧来提升随机梯度法的求解计算效率？首先我们回到经典随机梯度法，其迭代格式为

$$\mathrm{x}^{k+1} = \mathrm{x}^k - \alpha_k \nabla f_{i(k)}\left(\mathrm{x}^k\right),$$

注意到如果把方差减小算法中随机变量 $\mathbb{X}$ 定义为 $\mathbb{X} = \nabla f_{i(k)}\left(\mathrm{x}^k\right)$，那么

$$\mathbb{E}[\mathbb{X}] = \mathbb{E}\left[\nabla f_{i(k)}\left(\mathrm{x}^k\right)\right] = \frac{1}{n}\sum_{j=1}^{n}\nabla f_j\left(\mathrm{x}^k\right) = \nabla g\left(\mathrm{x}^k\right).$$

该随机变量 $\mathbb{X}$ 的期望为整个目标函数 $g(\mathrm{x})$ 在 x^k 的梯度，那么随机梯度法所用的梯度信息可以看作为对整体函数梯度的随机采样。既然可以看作为随机采样，我们希望在 $\mathbb{X}$ 的基础上建立一种新的随机变量进行梯度采样，这个新的随机变量依然是整体的无偏估计，但是比随机变量 $\mathbb{X}$ 有更小的方差，从而使得每一个的随机变量采样估计更加稳定 (波动更小)。此时根据方差减小技巧，引人 $\mathbb{Y} = \nabla f_{i(k)}(\tilde{\mathrm{x}})$，其中 $\tilde{\mathrm{x}}$ 为某一个固定参数，α 设定为 1，那么

$$\begin{aligned}\mathbb{Z}_1 &= \mathbb{X} - \mathbb{Y} + \mathbb{E}[\mathbb{Y}] \\ &= \nabla f_{i(k)}\left(\mathrm{x}^k\right) - \nabla f_{i(k)}(\tilde{\mathrm{x}}) + \mathbb{E}\left[\nabla f_{i(k)}(\tilde{\mathrm{x}})\right] \\ &= \nabla f_{i(k)}\left(\mathrm{x}^k\right) - \nabla f_{i(k)}(\tilde{\mathrm{x}}) + \frac{1}{n}\sum_{i=1}^{n} f_i(\tilde{\mathrm{x}}).\end{aligned}$$

进一步利用新定义的随机变量 $\mathbb{Z}_1$ 来设计新的方差减小随机梯度。综合以上信息，我们给出结合方差减小技巧的随机梯度法 (Stochastic variance reduced gradient method, SVRG)[SVRG] 如算法 (**??**) 所示。

算法 3 方差减小随机梯度法

输入: 初始值 $\tilde{\mathrm{w}} \in \mathbb{R}^d$
输出: 最优解 $\tilde{\mathrm{x}}$
for $i = 1, 2, \cdots$ **do**
  计算在 $\tilde{\mathrm{x}}$ 处整个目标函数的梯度信息

$$\nabla g(\tilde{\mathrm{x}}) = \frac{1}{n}\sum_{j=1}^{n}\nabla f_j(\tilde{\mathrm{x}});$$

  令 $\mathrm{x}^0 = \tilde{\mathrm{x}}$
  for $k = 0, \cdots, \ell - 1$ **do**
    选择 $i(k)$
    计算更新 x^{k+1}

$$\mathrm{x}^{k+1} = \mathrm{x}^k - \alpha_k\left[\nabla f_{i(k)}\left(\mathrm{x}^k\right) - \nabla f_{i(k)}(\tilde{\mathrm{x}}) + \nabla g(\tilde{\mathrm{x}})\right]; \tag{4.7}$$

  end
  更新 $\tilde{\mathrm{w}} = \mathrm{w}^{\ell}$
end

方差减小随机梯度法的核心是式 **(??)**，而算法通过两层迭代来实现方差减小。外层迭代一般较小，所以全梯度的计算通常会较小，不会占据主要计算。式 **(??)** 可以看作为 SVRG 的代表迭代格式。

4.4 随机梯度法的扩展讨论

首先，我们总结比较一下本章节介绍的几种随机梯度法。

$$\begin{cases} \text{SGD}^{\textbf{SGD}}: & \mathrm{x}^{k+1}=\mathrm{x}^{k}-\alpha_k \nabla f_{i(k)}\left(\mathrm{x}^{k}\right) \\ \text{SAG}^{\textbf{SAG}}: & \mathrm{x}^{k+1}=\mathrm{x}^{k}-\alpha_k \left[\frac{1}{n}\left(\nabla f_{i(k)}\left(\mathrm{x}^{k}\right)-\boldsymbol{y}_{i(k)}^{k}\right)+\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{y}_{i}^{k}\right] \\ \text{SVRG}^{\textbf{SVRG}}: & \mathrm{x}^{k+1}=\mathrm{x}^{k}-\alpha_k \left[\nabla f_{i(k)}\left(\mathrm{x}^{k}\right)-\nabla f_{i(k)}(\tilde{\mathrm{x}})+\frac{1}{n}\sum_{i=1}^{n} f_i(\tilde{\mathrm{x}})\right] \\ \text{SAGA}^{\textbf{SAGA}}: & \mathrm{x}^{k+1}=\mathrm{x}^{k}-\alpha_k \left[\nabla f_{i(k)}\left(\mathrm{x}^{k}\right)-\boldsymbol{y}_{i(k)}^{k}+\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{y}_{i}^{k}\right] \end{cases} \tag{4.8}$$

在机器学习中，最常用的优化算法就是 SGD$^{\textbf{SGD}}$，但是由于其每次迭代方向都是随机选择的，所以在下降方向会产生方差，导致在每次迭代（使用固定步长 α）时，最终是收敛不到最优值的。为了克服这一点，则需要让方差呈递减趋势下降，那么最终算法的将会以线性速度收敛于最优值。基于这种思想，学者提出了 SAG$^{\textbf{SAG}}$、SVRG$^{\textbf{SVRG}}$ 和 SAGA$^{\textbf{SAGA}}$ 这三种算法。SAG 算法$^{\textbf{SAG}}$ 在内存中为每个样本都维护一个旧的梯度 y_i，随机选择一个样本 i 来更新，并用 d 来更新 $\nabla f_{i(k)}\left(\mathrm{x}^{k}\right)$。如此，每次更新的时候仅仅需要计算一个样本的梯度，而不是所有样本的梯度。计算开销与 SGD$^{\textbf{SGD}}$ 无异，但是内存开销要大得多。文献$^{\textbf{SAG}}$ 中已经证明 SAG$^{\textbf{SAG}}$ 是一种线性收敛算法，这个速度远比 SGD$^{\textbf{SGD}}$ 快。SVRG$^{\textbf{SVRG}}$ 的算法思路是，每过一段时间计算一次所有样本的梯度 $\frac{1}{n}\sum_{i=1}^{n} f_i(\tilde{\mathrm{x}})$。相对于 SAG$^{\textbf{SAG}}$ 来说，不需要在内存中为每个样本都维护一个梯度，也就是说节省了内存资源。SAGA 算法$^{\textbf{SAGA}}$ 是 SAG$^{\textbf{SAG}}$ 算法的一个加速版本，和 SAG$^{\textbf{SAGA}}$ 一样，它既不在一个循环里面操作，也不计算批量梯度 (除了在初始点)，在每次迭代中，它都计算随机向量 x^k 作为前期迭代中随机梯度的平均值。两者的区别是，SAGA$^{\textbf{SAGA}}$ 是无偏的方差减小随机梯度方法，而 SAG 算法$^{\textbf{SAG}}$ 则是有偏的。

4.5 随机梯度法的求解应用

例 4.1 (矩阵分解模型). 以最基本的矩阵分解模型 $r_{ui}=q_i^T p_u$ 为例，损失函数可以定义为 $J\left(p_u,q_i\right)=\frac{1}{2}\left(\sum\left(r_{ui}-q_i^T p_u\right)^2+\lambda\left(\|q_i\|^2+\|p_u\|^2\right)\right)$，其中 λ 是正则化系数，这里把对 p_u 和 q_i 的正则化系数设置为相同值（也可以为不同值，不使用括号即可）。
下面求损失函数对 p_u 的偏导：

$$\frac{\partial J\left(p_u\right)}{\partial p_u}=\left(r_{ui}-q_i^T p_u\right)\left(-q_i^T\right)+\lambda p_u.$$

常令

$$e_{ui} = r_{ui} - q_i^T p_u,$$

因此

$$p_u = p_u - \eta \cdot \frac{\partial J(p_u)}{\partial p_u} = p_u + \eta \left(e_{ui} \cdot q_i - \lambda \cdot p_u\right).$$

其中，η 表示学习速率。

同理

$$q_i = q_i - \eta \cdot \frac{\partial J(q_i)}{\partial q_i} = q_i + \eta \left(e_{ui} \cdot p_u - \lambda \cdot q_i\right).$$

5. 临近梯度下降法

本节引入临近算子与临近梯度的概念，分析临近梯度下降法，并使用该方法求解具体的凸优化问题。

5.1 凸优化模型

考虑凸优化问题 $\min\limits_{x\in\mathbb{R}^n} f(x) := g(x) + h(x)$，其中函数 $g(x)$ 是可微的且 Lipschitz 连续，而 $h(x)$ 则可能是连续但不处处可微的函数。例如，在对优化的函数引入正则项时，就属于上述的凸优化模型，其中 $h(x) = \|x\|_1$ 即 l_1 范数。对于这样的模型，可以采用临近梯度下降法求解最优解。

在应用临近梯度下降法进行优化时，通常对 Lipschitz 连续的部分即 $g(x)$ 求梯度，但对于不处处可微的 $h(x)$ 部分在无法求解梯度时，采用求临近梯度的方式代替。

5.2 临近算子与临近梯度

在对 $h(x)$ 求临近梯度[2014Prox] 前，首先引入临近算子的概念，用于处理非光滑函数，计算求解一个距离 x 最近的点使得 $h(x)$ 最小。

定义 5.1. 对于一个凸函数 $h(x)$，定义它的临近算子为

$$\text{prox}_{\text{h}}(x) = \underset{u\in\text{dom}\, h}{\text{argmin}}\{h(u) + \frac{1}{2}\|u - x\|^2\}. \tag{5.1}$$

如果想进一步基于临近算子的概念定义临近梯度，并用于优化问题的求解，必须证明上述定义的临近算子唯一存在，这样对于一个给定的凸优化问题，其最优解才是唯一的。

定理 5.2. 如果 $h(x)$ 是一个定义在 $\mathbb{R}^n$ 上的适当的闭凸函数，则对任意的 $x \in \mathbb{R}^n$，$\text{prox}_{\text{h}}(x)$ 的值唯一存在

证明: 假设 $h(x)$ 至少在定义域内存在一点存在次梯度 $\boldsymbol{g}$，由次梯度的定义可以得到

$$h(u) \geq h(v) + \langle \boldsymbol{g}, u - v\rangle, v \in \text{dom}h\,. \tag{5.2}$$

定义辅助函数

$$m(u) = h(u) + \frac{1}{2}\|u - x\|^2. \tag{5.3}$$

可以得到

$$m(u) \geq h(v) + \langle \boldsymbol{g}, u - v\rangle + \frac{1}{2}\|u - x\|^2. \tag{5.4}$$

因此，$m(u)$ 是适当闭函数，由 Weierstrass 定理可知其存在最小值。

同时，注意到 $m(u)$ 是强凸函数，由强凸函数的性质可知，$m(u)$ 最小值唯一。

综上所述，$\text{prox}_{\text{h}}(x)$ 的值存在且唯一。 □

以 l_1 范数为例给出其临近算子的计算，这常被用于作为优化问题的 l_1 正则项，例如在 Lasso 问题中也将运用 l_1 范数的临近算子计算。

问题 5.3. 求解 l_1 范数的临近算子，即求

$$\text{prox}_{\text{h}}(x), h(x) = t\|x\|_1, t > 0. \tag{5.5}$$

解: 给定 l_1 范数 $h(x) = t\|x\|_1, t > 0$，按照定义**??**可以得到其临近算子的形式为

$$\text{prox}_{\text{h}}(x) = \underset{u\in\mathbb{R}^n}{\text{argmin}} \begin{cases} tu + \frac{1}{2}(u-x)^2, u > 0 \\ -tu + \frac{1}{2}(u-x)^2, u \le 0 \end{cases} \tag{5.6}$$

进一步考虑，当 $x > t$ 时，$\text{prox}_{\text{h}}(x) = x - t$；当 $x < -t$ 时，$\text{prox}_{\text{h}}(x) = x + t$；当 $-t \le x \le t$ 时，$\text{prox}_{\text{h}}(x) = 0$，即 $h(x)$ 唯一不可微的点。

综上所述，可以引入符号函数简化上述分类，即

$$\text{prox}_{\text{h}}(x) = \text{sgn}(x)\max\{|x| - t, 0\}. \tag{5.7}$$

其中，$\text{sgn}(x)$□□□□□□□□□□□□

定义: 对于 $\text{sgn}(x)$，定义

$$\text{sgn}(x) = \begin{cases} 1, & x > 0, \\ 0, & x = 0, \\ -1, & x < 0. \end{cases} \tag{5.8}$$

$\text{sgn}(x)$ 函数的图像如下图**??**所示。

由定理**??**可进一步定义临近梯度的概念，用于作为梯度的扩展概念，处理一个可分解的函数中非光滑的部分，而对于光滑的部分则使用梯度计算。

定义 5.5. 对于 $f(x) := g(x) + h(x)$，其中 $g(x)$ 是连续可微的且 Lipschitz 连续，$h(x)$ 是连续但不处处可微的，定义它的临近梯度为

$$\tilde{\nabla} f(x) = x - \text{prox}_{\alpha\text{h}}(x - \alpha\nabla g(x)), \tag{5.9}$$

其中，$\alpha > 0$，在临近梯度下降法中的意义同梯度下降法中的步长。

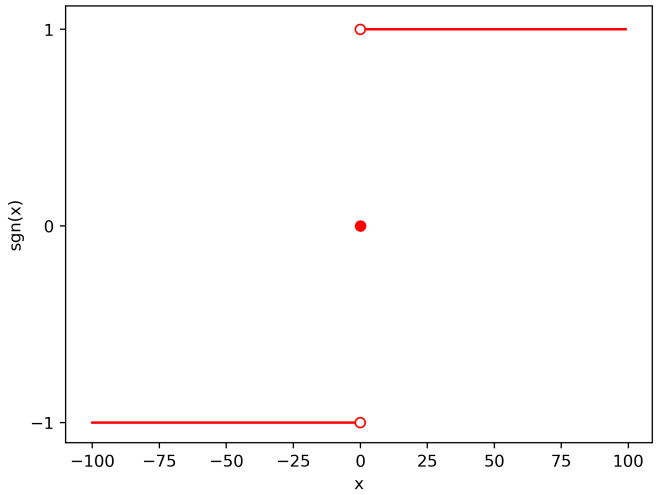

图 5-1: sgn(x) 函数图像

显然，当非光滑的部分 $h(x)=0$ 时，临近梯度 $\tilde{\nabla} f(x)=\alpha \nabla g(x)=\alpha \nabla f(x)$ 即梯度。这表明了临近梯度的定义是梯度的扩展，也说明了梯度是临近梯度的特殊情况。

5.3 临近梯度下降法

对于凸优化问题 $\min\limits_{x\in\mathbb{R}^n} f(x):=g(x)+h(x)$，其中，$g(x)$ 是可微且 Lipschitz 连续的函数，而 $h(x)$ 则是不处处可微的凸函数。

针对光滑的部分 $g(x)$ 采用计算梯度的方式，而对于非光滑的部分 $h(x)$ 采用计算临近算子的方式，下面给出迭代公式

$$\begin{cases} y^{k+1}=x^k-\alpha_k \nabla g(x^k) \\ x^{k+1}=\text{prox}_{\alpha_k h}(y^{k+1}) \end{cases} \tag{5.10}$$

或者将两次迭代合并写成

$$x^{k+1}=\text{prox}_{\alpha_k h}(x^k-\alpha_k \nabla g(x^k)), \tag{5.11}$$

其中，α_k 表示第 k 次迭代时算法选取的步长，通常 $0<\alpha<1$；x^k 表示第 k 次的迭代。

由定义**??**对临近梯度的定义可知，临近梯度下降法其实是计算了使得 $\tilde{\nabla} f(x)=0$ 的解，并将其作为问题的最优解。

结合迭代公式**??**与初始化设置，可以将临近梯度下降法总结为如下算法。

算法 4 临近梯度下降法

输入: $f(x) := g(x) + h(x)$，初始化设置 $k = 0$
输出: 最优解 x^*
while 未收敛 **do**
  $x^{k+1} = \text{prox}_{\alpha_k h}(x^k - \alpha_k \nabla g(x^k))$
  $k \leftarrow k + 1$
end

5.4 算法分析

首先，对于问题**??**，临近梯度下降法实际是对非光滑部分 $g(x)$ 在 x^k 处的 Taylor 展开再加上二次项，同时保留非光滑部分 $h(x)$，对该展开式求极小来作为每次迭代的近似。

$$\begin{aligned} x^{k+1} &= \underset{u\in \text{dom}\, h}{\text{argmin}}\{\alpha_k h(u) + \frac{1}{2}\|u - (x^k - \alpha_k \nabla g(x^k))\|^2\} \\ &= \underset{u\in \text{dom}\, h}{\text{argmin}}\{h(u) + g(x^k) + \langle \nabla g(x^k), u - x^k\rangle + \frac{1}{2\alpha_k}\|u - x^k\|^2\}. \end{aligned} \tag{5.12}$$

其次，临近梯度下降法对光滑的部分做显式的梯度下降，对非光滑部分根据其性质与结构使用临近算子做隐式的梯度下降进行求解。这相比一般的梯度下降法能够求解更多的问题，也是临近梯度下降法的优势。

最后，考虑算法**??**的收敛性分析。

定理 5.6. 假设给定步长为常数，即 $\alpha_k = \alpha \in (0, \frac{1}{L}]$，后记作 α，其中 L 表示 Lipschitz 连续的部分 $g(x)$ 满足 $\|\nabla g(x) - \nabla g(y)\| \leq L\|x - y\|, \forall x, y \in \mathbb{R}^n$，迭代点 x^k 处的函数值 $f(x^k)$ 以 $O(\frac{1}{k})$ 的速率收敛到最优解 x^*

证明: 由下降引理可得

$$g(y) \leq g(x) + \langle \nabla g(x), y - x\rangle + \frac{L}{2}\|x - y\|^2, \forall x, y \in \mathbb{R}^n.$$

令 $y = x - \alpha\tilde{\nabla} g(x)$，有

$$g(x - \alpha\tilde{\nabla} g(x)) \leq g(x) - \alpha\langle \nabla g(x), y - x\rangle\tilde{\nabla} g(x) + \frac{\alpha^2 L}{2}\|\tilde{\nabla} g(x)\|^2.$$

由 $0 < t \leq \frac{1}{L}$ 可知，

$$g(x - \alpha\tilde{\nabla} g(x)) \leq g(x) - \alpha\langle \nabla g(x), y - x\rangle\tilde{\nabla} g(x) + \frac{\alpha}{2}\|\tilde{\nabla} g(x)\|^2.$$

由 $g(x), h(x)$ 为凸函数，对 $\forall z \in \mathrm{dom}\, f$，有

$$g(x) \leq g(z) - \langle \nabla g(x), z - x \rangle.$$

整理可得，对 $\forall z \in \mathrm{dom}\, f$，有

$$f(x - \alpha \tilde{\nabla} g(x)) \leq f(z) + \langle \tilde{\nabla} g(x), z - x \rangle - \frac{\alpha}{2} \|\tilde{\nabla} g(x)\|^2.$$

记第 $k+1$ 次迭代为 $x^{k+1} = x^k - \alpha \tilde{\nabla} g(x^k)$，
令 $z = x^*$ 为问题的最优解，有

$$\begin{aligned} f(x^{k+1}) - f(x^*) &\leq \langle \tilde{\nabla} g(x^k), x^k - x^* \rangle - \frac{\alpha}{2} \|\tilde{\nabla} g(x^k)\|^2 \\ &= \frac{1}{2\alpha} (\|x^k - x^*\|^2 - \|x^k - x^* - \alpha \tilde{\nabla} g(x^k)\|^2) \\ &= \frac{1}{2\alpha} (\|x^k - x^*\|^2 - \|x^{k+1} - x^*\|^2). \end{aligned} \tag{5.13}$$

将前 k 次迭代的结果累加代入上式可得，

$$\begin{aligned} \sum_{i=1}^{k} (f(x^i - f(x^*))) &\leq \frac{1}{2\alpha} \sum_{i=1}^{k} (\|x^{i-1} - x^*\|^2 - \|x^i - x^*\|^2) \\ &= \frac{1}{2\alpha} (\|x^0 - x^*\|^2 - \|x^k - x^*\|^2) \\ &\leq \frac{1}{2\alpha} \|x^0 - x^*\|^2. \end{aligned} \tag{5.14}$$

其中，x^0 同算法**??**中的初始化设置。
因此，

$$f(x^k - f(x^*)) \leq \frac{1}{2k\alpha} \|x^0 - x^*\|^2. \tag{5.15}$$

综上所述，迭代点 x^k 处的函数值 $f(x^k)$ 以 $O(\frac{1}{k})$ 的速率收敛到最优解 x^*。 □

5.5 应用

以 Lasso 问题为例，使用临近梯度下降法求解该优化问题。

问题 5.7. 求解 Lasso 问题，即求

$$\min_x f(x) := \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \mu\|\boldsymbol{x}\|_1. \tag{5.16}$$

解: 考虑凸函数 $f(\boldsymbol{x})$ 可以分解为光滑的部分 $g(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$，与非光滑的部分 $h(\boldsymbol{x}) = \mu\|\boldsymbol{x}\|_1$，

对光滑部分求梯度可以得到

$$\nabla g(\boldsymbol{x}) = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}).$$

由**??**，对非光滑部分求临近算子可以得到

$$\text{prox}_{\alpha_k h}(\boldsymbol{x}) = \text{sgn}(\boldsymbol{x})\max\{|\boldsymbol{x}| - \alpha_k\mu, 0\},$$

其中，α_k 表示第 k 次迭代的步长。

由迭代公式**??**，可以得到

$$\begin{aligned}
\boldsymbol{x}^{k+1} &= \text{prox}_{\alpha_k h}(\boldsymbol{x}^k - \alpha_k\nabla g(\boldsymbol{x}^k)) \\
&= \text{prox}_{\alpha_k h}(\boldsymbol{x}^k - \alpha_k\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{b})) \\
&= \text{sgn}(\boldsymbol{x}^k - \alpha_k\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{b}))\max\{\|\boldsymbol{x}^k - \alpha_k\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{b})\| - \alpha_k\mu, 0\}
\end{aligned} \tag{5.17}$$

使用 Python 对该问题求解，可以得到结果如图**??**所示，代码在附录中给出。其中，alpha 表示算法**??**中的学习率。可以看出，随着学习率的增加，算法收敛的速度也越快，特别地，在学习率较小的时候，算法未能在给定的 300 个 epoch 内收敛。

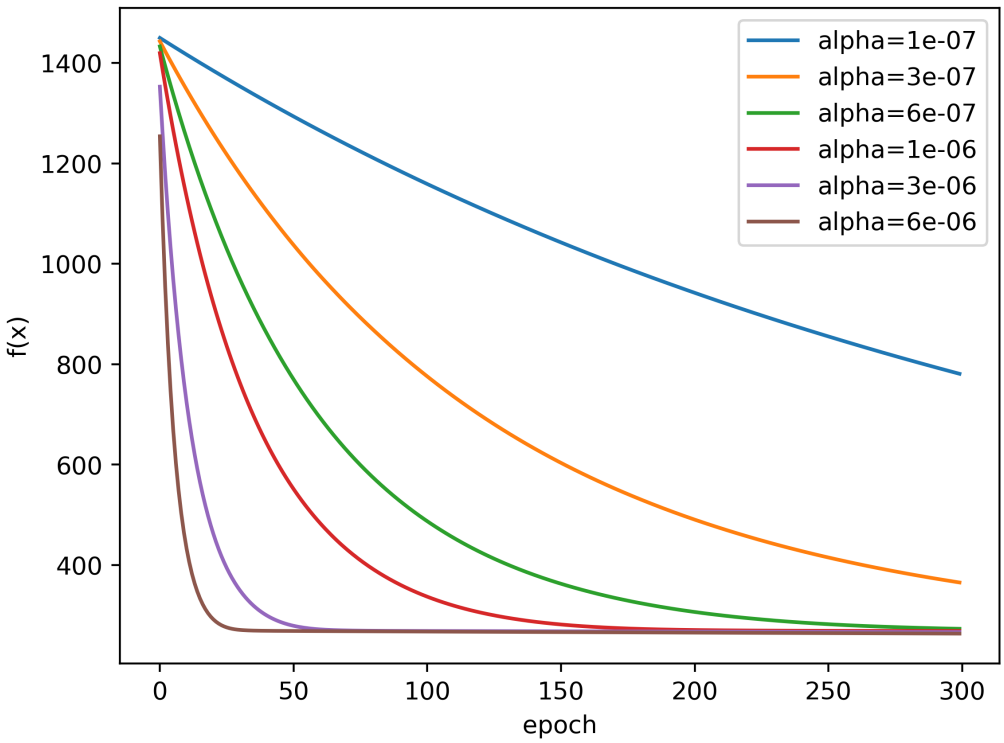

图 5-2: Python 实现临近梯度下降法求解 Lasso 问题

6. 牛顿迭代法

6.1 牛顿迭代法的定义

牛顿迭代法（Newton's method）是牛顿在 17 世纪提出的一种在实数域和复数域上近似求解方程的方法。对于大多数方程而言求精确根非常困难，甚至存在不可解情况，从而求解这类方程的问题转为寻找方程的近似根。牛顿迭代法使用函数 $f(x)$ 的泰勒级数的前面几项来寻找方程 $f(x)=0$ 的根。假设目标函数 $f(x)$ 在 x 处二阶可微，且当前的迭代点为 x_k，那么在出的泰勒展开时为

$$f(x_k+d)=f_k+g_k^Td+\frac{1}{2}d^TG_kd+(\|d\|_2^2). \tag{6.1}$$

由于 $d=x-x_k$，则式**??**可转为式**??**表达

$$\min q_k(d)=f_k+g_k^Td+\frac{1}{2}d^TG_kd. \tag{6.2}$$

若 G_k 正定，则方程组 $G_kd=-g_k$ 的 $d_k=-G_k^{-1}g_k$ 解为以上问题的唯一解。我们称上述方程组为牛顿方程组，解得的方向 d_k 为牛顿方向。用牛顿方向作为迭代方向的最优化方法称为牛顿方法。特别的，全步长 $\alpha_k=1$ 的牛顿方法称为基本牛顿方法。

算法 5 牛顿迭代法的算法

输入: 初始点 $x_0\in\mathbb{R}^n,\varepsilon>0,k=0$
输出: 最优解 x_{k+1}
while 未收敛 **do**
  $d_k=-G_k^{-1}g_k$
  $x_{k+1}=x_k+\alpha_kd_k$
end

6.2 牛顿迭代法算例

例 6.1.

$$\text{≽⊐⋉}f(x)=\frac{3}{2}x^2+\frac{1}{2}y^2-xy-2x,x_0=-2,y_0=4.$$

解:

$$\nabla f(x,y)=\begin{pmatrix}3x-y-2\\-x+y\end{pmatrix}$$

, 由于 $g_k = \nabla f(x, y)$，则

$$G = \nabla^2 f(x, y) = \begin{pmatrix} 3 & -1 \\ -1 & 1 \end{pmatrix}$$

. 将 $x_0 = -2, y_0 = 4$ 代入 g_k，得到 $g_0 = \begin{pmatrix} -12 \\ 6 \end{pmatrix}$, 同时因为 G 为正定矩阵，则存在逆矩阵

$$G^{-1} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{3}{2} \end{pmatrix}$$

. 将 G^{-1} 代入牛顿迭代公式 $x_{k+1} = x_k - G^{-1} g_K$，得到

$$x_1 = x_0 - G^{-1} * g_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

. 迭代直到满足终止条件。 ■

6.3 算法的收敛性及其优缺点

基本牛顿方法具有二阶收敛速度，但只有迭代点充分接近 x^* 时，才能保证其收敛性

定理 6.2. 设 $f(x) \in \mathbb{R}^2, f(x)$ 的 Hesse 矩阵 $G(x)$ 满足 Lipschitz 条件，即存在 $\beta > 0$，对任给的 x 与 y，有 $\|G(x) - G(y)\| \leq \beta \|x - y\|$。若 x_0 充分接近 $f(x)$ 的局部极小点 x^*，且 G^* 正定，则牛顿方法对所有的 k 有定义，并以二阶收敛速度收敛

牛顿迭代法的 **优点**是：当 x_0 充分接近问题的极小点 x^* 时，方法拥有二阶收敛速度；相比于梯度下降法的一阶收敛有着更快的收敛速度。由于牛顿法使用二次曲面去拟合当前所处位置的局部曲面，而梯度下降法是用一个平面去拟合当前的局部曲面，通常情况下，二次曲面的拟合会比平面更好，所以牛顿法选择的下降路径会更符合真实的最优下降路径。**缺点**是：当 x_0 没有充分接近问题的极小点 x^* 时，G_k 会出现不正定或奇异的情况，使得 x_k 不能收敛到 x^*，甚至无法迭代下去；即使 G_k 正定，f_k 仍未必单调下降；每次使用牛顿法进行计算的时候都需要计算 Hesse 矩阵，迭代计算量大。

6.4 牛顿迭代法的改进

为克服牛顿迭代法的缺点，可以通过以下几种方法来改进：

6.4.1 阻尼牛顿方法

阻尼牛顿法是带一维搜索的牛顿方法。牛顿法缺点中，确定了迭代方向之后，迭代步长默认为 1，但是这个迭代方向并不一定是朝着函数值下降的方向。所以阻尼牛顿法为了解决这个问题，采取的做法是确定了迭代方向（和牛顿法一样的迭代方向）之后，还需要在该方向做一维搜索，寻找使得在该迭代方向上最优的迭代步长。即 $x_{k+1} = x_k + \alpha_k d_k$，其中 α_k 是一维搜索的结果。其优点是：对于正定的 G_k，f_k 单调下降；即使 x_0 离 x^* 稍远，x_k 仍可能收敛到 x^*。对于严格凸函数，采用 Wolfe 准则的阻尼牛顿法具有全局收敛性，这一点改善了牛顿迭代法的局部收敛性质。

6.4.2 混合方法

为解决牛顿迭代法在迭代过程中可能出现的 Hesse 矩阵奇异、不正定或牛顿方向与 g_k 几乎正交的情形，我们可以采用混合方法。这里我们考虑的方法是将负梯度方法的混合，该方法采用牛顿方向，但在 Hesse 矩阵 G_k 奇异或 d_k 与 g_k 几乎正交时，采用负梯度方向；在 G_k 负定，但 G_k^{-1} 存在时，取 $d_k = G_k^{-1} g_k$。具体算法如下：

算法 6 混合算法的算法

输入: 初始点 $x_0 \in \mathbb{R}^n, \varepsilon > 0, k = 0$
输出: 最优解 x_{k+1}
while 未收敛 **do**
  if G_k 非奇异 **then**
    由牛顿方程求得 d_k
    if $g_k^T d_k > \varepsilon_1 \|g_k\| \|d_k\|$ **then**
      $d_k = -d_k$
    end
    else if $|g_k^T d_k| \leq \varepsilon_1 \|g_k\| \|d_k\|$ **then**
      $d_k = -g_k$
    end
  end
  else
    $d_k = -g_k$
  end
  线搜索求 $\alpha_k, x_{k+1} = x_k + \alpha_k d_k, k = k + 1$
end

7. 拟牛顿方法

牛顿迭代法的缺点是每部都需要计算 Hesse 矩阵 G_x，G_x 可能奇异或不正定，所以构造了拟牛顿法，既不需要计算二阶偏导数，同时又有较快的收敛速度。拟牛顿迭代法的基本思想是利用 x_k，x_{k+1} 及其一阶信息构造一个正定的矩阵 $B_{k+1} \approx G_{k+1}$，此时产生的下降

方向 d_{k+1} 满足:$B_{k+1}d_{k+1} = -g_{k+1}$. 利用上述同样的信息构造一个正定的矩阵 $H_{k+1} \approx G_{k+1}^{-1}$ 此时产生的下降方向 d_{k+1} 满足:$d = -H_{k+1}g_{k+1}$.

其中 $B_{k+1} \approx G_{k+1}$，满足如下拟牛顿方程:$B_{k+1}s_k = y_k$.

若记 $H_{k+1} = B_{k+1}^{-1}$，则 H_{k+1} 应满足 $H_{k+1}y_k = s_k$

下面以矩阵 H_k 为例，给出一般拟牛顿法的结构

算法 7 拟牛顿方法的算法

输入: 初始点 $x_0 \in \mathbb{R}^n$，对称正定阵 $H_0 \in \mathbb{R}^{n\times n}, \varepsilon > 0, k = 0$
输出: 最优解 x_{k+1}
while 未收敛 **do**
  $d_k = -H_k g_k$
  沿方向 d_k 进行线搜索求 $\alpha_k > 0$，$x_{k+1} = x_k + \alpha_k d_k$
  修正 H_k 得 H_{k+1}，使 H_{k+1} 满足 $H_{k+1}y_k = s_k, k = k + 1$
end
注：H_k 的选取和修正是关键，通常 $H_0 = I$，则 $d_0 = -g_0; H_{k+1} = H_k + \Delta H_k$

7.1 各种拟牛顿方法及其优缺点

7.1.1 SR1

1980Curvilinear SR1 方法也即对对称秩 1 方法更新，根据 x_k 处的信息得到一个修正量 ΔH_k 来直接加上 H_k 来更新，也就是如下的式子:

$$H_{k+1} = H_k + \Delta H_k \tag{7.1}$$

由于我们希望 $\Delta H_{k+1} \approx \nabla^2 f(x_{k+1})^{-1}, \Delta H_k \approx \nabla^2 f(x_k)^{-1}$,所以有 $\Delta H_k \approx \nabla^2 f(x_{k+1})^{-1} - \nabla^2 f(x_k)^{-1}$

由于 $\nabla^2 f(x_{k+1})^{-1}$ 和 $\nabla^2 f(x_k)^{-1}$ 都是对称矩阵，所以 ΔH_k 也是对称矩阵。SR1 算法将 ΔH_k 设为 βuu^T，则迭代式为:

$$H_{k+1} = H_k + \beta uu^T, \beta \in \mathbb{R}, u \in \mathbb{R}^n \tag{7.2}$$

优点: SR1 算法可以使得最终选择的 B_k 会一步步的接近于函数的 Hesse 矩阵，它之后会越来越逼近 Hesse 矩阵，可以恢复牛顿法的二次局部收敛速度。不需要一维搜索，而具有二次终止性；具有遗传性：$H_i y_j = s_j, j < i$.

缺点: 矩阵 B_k 不保正定，如果使用线搜索会导致无法找到下降方向的问题，使得 SR1 校正不保持迭代矩阵 H_k 的正定性，仅当 $(s_k - H_k y_k)^T y_k > 0$ 时，SR1 校正才具有正定性。

另一方面，矩阵可以很好的逼近 Hesse 矩阵，但是线搜索每一次只会利用到一个方向上的 Hesse 矩阵的信息

7.1.2 DFP

1963A DFP 方法是在 SR1 的方法基础上对于 ΔH 进行进一步更新, 相对于 SR1 令 $\Delta H_k = \beta uu^T$，DFP 提供了更大的自由度，令 $\Delta H_k = \beta uu^T + \gamma vv^T$，则此时 H_k 的迭代更新式为:

$$H_{k+1} = H_k + \beta uu^T + \gamma vv^T \tag{7.3}$$

优点: DFP 法只需计算一阶偏导数, 无需计算二阶偏导数及其逆矩阵, 对于二次函数具有二次终止性; 对目标函式的初始点选择均无严格要求, 收敛速度快; 为提高实际计算的稳定性，除提高一维搜寻的精度外，通常还将进行 n 次迭代作为一个循环，并以上一循环的终点作为起点继续进行循环叠代，使得其对于一般函数校正保持正定性；当采用精确线搜索时，对于凸函数具有总体收敛性。

缺点: DFP 方法具有数值不稳定，有时产生数值上的 Hesse 近似。

7.1.3 BFGS

1989On BFGS 在 DFP 的基础上，对于 $\nabla^2 f(x_k)$ 进行近似，令 $B_k = \nabla^2 f(x_k)$，得到

$$B_{k+1} = B_k + \beta uu^T + \gamma vv^T \tag{7.4}$$

优点: 具有 DFP 校正所有的各种性质, 同时克服了 DFP 方法的上述缺点; 当采用 Goldstein 或 Wolfe 线搜索时，BFGS 公式还具有总体收敛性；数值执行中，BFGS 公式也优于 DFP 公式，尤其是它常常能与低精度线搜索方法一起使用。

7.2 算例分析

例 7.1. 考虑问题

$$minf(x) = \sum_{i=1}^{m} r_i^2(x), n = 6, m \geq n, \tag{7.5}$$

其中，$r_i(x) = x_3 e^{-t_i x_1} - x_4 e^{-t_i x_2} + x_6 e^{-t_i x_5} - y_i, t_i = 0.1i, y_i = e^{-t_i} - 5e^{-10t_i} + 3e^{-4t_i}$.

对例**??**而言，DFP 的方法的有效性明显低于 BFGS 方法。在表**??**中，ite 表示迭代次数，feva 表示调用函数的次数可以看到 DFP 方法大部分情况下的迭代次数都超过 BFGS 方法，

表 7-1: $\sigma = 0.1$ 时,SR1,BFGS 与 DFP 三种方法的迭代次数和函数调用次数

m	SR1		BFGS		DFP	
	ite	feva	ite	feva	ite	feva
6	113	746	14	102	314	1935
7	52	481	49	303	18	128
8	172	1251	27	126	182	874
9	17	111	27	137	197	1233
10	11	68	88	460	128	686
11	28	186	25	110	187	959
12	22	130	24	98	9	71
13	16	119	28	124	24	112

表 7-2: $\sigma = 0.1$ 时,SR1,BFGS 与 DFP 三种方法的所得解点 x_k 处的 $\|g_k\|_\infty$

m	SR1	BFGS	DFP
6	7.05e-5	2.74e-0	1.20e-1
7	4.54e-6	1.09e-4	3.85e-1
8	6.23e-5	1.04e-6	1.54e-1
9	7.59e-9	1.81e-5	6.35e-1
10	9.71e-6	2.14e-5	9.44e-1
11	4.57e-3	2.04e-6	6.93e-3
12	5.87e-4	8.40e-2	1.14e-1
13	2.50e-1	4.13e-7	6.14e-6

而调用次数 DFP 方法多数情况下也多于 BFGS 方法。而 SR1 方法所需的函数调用次数和所得结果的精确度与 BFGS 方法更相近，但 SR1 方法不如 BFGS 方法稳定。

实际上，DFP 方法对于许多问题都有类似的数值表现。在表**??**中，DFP 的解梯度值远超过 SR1 和 BFGS 的方法同时注意到，BFGS 方法在 m=6,12 时，解点的梯度值 $\|g_k\|_\infty$ 不小，这是由于在 $k-1$ 步迭代，找到的 x_k 不能使函数值有足够的下降，从而导致 $f_{k+1}-f_k$ 满足精度要求，停止迭代造成的。

8. 共轭梯度法

8.1 线性共轭梯度法

一般的共轭梯度法由线性共轭梯度法推导而来，所以我们首先考虑关于正定二次函数的共轭梯度法，先有以下定义:

设 G 是 n 阶对称正定矩阵。若 R^n 中两个非零向量 $d_i, d_j (i \neq j)$ 满足

$$d_i^T G d_j = 0, \qquad i, j = 0, ..., n-1, i \neq j \tag{8.1}$$

则称 d_i 和 d_j 是共轭方向。

若 R^n 中 n 个非零向量 $d_0, ..., dd_{n-1}$ 满足

$$d_i^T G d_j = 0, \qquad i, j = 0, ..., n-1, i \neq j \tag{8.2}$$

则称它们为 G 的两两共轭方向，或称这个向量组是共轭的。

共轭方向是根据正定二次函数 $f(x) = \frac{1}{2}x^T G x + b^T x$ 的梯度来构造的，所以我们可以由此导出共轭梯度法的迭代方向 $d_k = -g_k + \beta_{k-1} d_{k-1}$. 因为对二次正定函数有 $g_k^T g_{k-1} = 0$，可以得到 $\beta_{k-1} = \dfrac{g_k^T g_k}{g_{k-1}^T g_{k-1}}$

算法 8 线性共轭梯度法的算法

输入: $x_0 \in \mathbb{R}^n, d_0 = -g_0, \varepsilon > 0$，初始化设置 $k = 0$
输出: 最优解 x_{k+1}
while 未收敛 **do**
  $\alpha_k = -\dfrac{g_k^T d_k}{d_k^T G d_k}$
  $x_{k+1} = x_k + \alpha_k d_k$
  $d_{k+1} = -g_{k+1} + \beta_k d_k, k = k + 1$
end

8.2 非线性共轭梯度法

由线性共轭梯度法推广而来，迭代方向与线性共轭梯度相同，由以下两个不同的非线性共轭梯度公式，可以给出非线性共轭梯度方法的算法步骤。

FR 公式:

$$\beta_{k-1}^{FR} = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}} \tag{8.3}$$

PRP 公式:

$$\beta_{k-1}^{PRP} = \frac{g_k^T(g_k - g_{k-1})}{g_{k-1}^T g_{k-1}} \tag{8.4}$$

算法 9 非线性共轭梯度法的算法

输入: $x_0 \in \mathbb{R}^n, d_0 = -g_0, \varepsilon > 0$，初始化设置 $k = 0$
输出: 最优解 x_{k+1}
while 未收敛 **do**
  一维线搜索求 α_k
  $x_{k+1} = x_k + \alpha_k d_k$
  $\beta_k, d_{k+1} = -g_{k+1} + \beta + kd_k, k = k + 1$
end

8.2.1 非线性共轭梯度法 FR 的算例

例 8.1.

$$\text{求}f(x) = \frac{3}{2}x_1^2 + \frac{1}{2}x_2^2 - x_1x_2 - 2x_1, x_0 = (0,0)^T.$$

解:

$$g(x) = \begin{pmatrix} 3x_1 - x_2 - 2 \\ -x_1 + x_2 \end{pmatrix}, G(x) = \begin{pmatrix} 3 & -1 \\ -1 & 1 \end{pmatrix}.$$

. 因 $g_0 = (-2,0)^T$，故取 $d_0 = (2,0)^T$，从 x_0 出发，沿 d_0 作一维搜索，即求

$$\text{求}f(x_0 + \alpha_0 d_0) = 6\alpha^2 - 4\alpha$$

的极小点，得步长 $\alpha_0 = \frac{1}{3}$。于是得到

$$x_1 = x_0 + \alpha_0 d_0 = (\frac{2}{3}, 0)^T, g_1 = (0, -\frac{2}{3})$$

. 由 FR 公式得

$$\beta_0^{FR} = \frac{g_1^T g_1}{g_0^T g_0} = \frac{1}{9}$$

, 故

$$d_1 = -g_1 + \beta_0 d_0 = (\frac{2}{9}, \frac{2}{3})^T$$

. 从 x_1 出发，沿 d_1 作一维搜索，求

$$\text{求}f(x_1 + \alpha_1 d_1) = \frac{4}{27}\alpha^2 - \frac{4}{9}\alpha + \frac{2}{3}$$

的极小点，解得 $\alpha_1 = \frac{3}{2}$，于是 $x_2 = x_1 + \alpha_1 d_1 = (1,1)^T$。此时 $d_2 = (0,0)^T$，故 $x_{k+1} = (1,1)^T, f_{k+1} = -1$。 ■

8.2.2 非线性共轭梯度法的理论结果

1992Global

定理 8.2. 对于 FR 方法，若 α_k 由强 Wolfe 准则得到，且 $\sigma \in (0, \frac{1}{2})$，则 d_k 满足

$$-\sum_{j=0}^{k-1}\sigma^j \le \frac{g_k^T d_k}{\|g_k\|^2} \le -2 + \sum_{j=0}^{k-1}\sigma^j, k = 1, ... \tag{8.5}$$

其中 d_k 是下降方向

证明: 通过数学归纳法进行证明。由于**??**这个结果显然适用于 k=1，因为所有三个项都是-1。假设**??**对某些 k>1 成立，这意味着 $< gk \square dk >< 0$，因为

$$-2 + \sum_{j=0}^{k-1}\sigma^J < -2 + \frac{1}{1-\sigma} = \frac{2\sigma - 1}{1-\sigma} < 0,$$

在 $\sigma < \frac{1}{2}$ 时，根据 FR 公式，我们得到

$$\frac{g_{k+1}^T d_{k+1}}{\|g_{k+1}\|^2} = -1 + \beta_{k+1}\frac{g_{k+1}^T d_k}{\|g_{k+1}\|^2} = -1 + \frac{\beta_{k+1}}{\beta_{k+1}^{\mathbb{FR}}}\frac{g_{k+1}^T d_k}{\|g_k\|^2}$$

. 根据线搜索条件，我们得到

$$|\beta + k + 1(g_{k+1}^T d_k)| \le -\sigma||\beta + k + 1|(g_k^T d_k)$$

, 将上式与式**??**联立，得到

$$-1 + \sigma\frac{|\beta_{k+1}|}{\beta_{k+1}^{\mathbb{FR}}}\frac{g_k^T d_k}{\|g_k\|^2} \le \frac{g_{k+1}^T d_{k+1}}{\|g_k\|^2} \le -1 - \sigma\frac{|\beta_{k+1}|}{\beta_{k+1}^{\mathbb{FR}}}\frac{g_k^T d_k}{\|g_k\|^2}$$

. 引入定理**??**中的左侧假设条件，我们可以得到

$$-1 - \sigma\frac{|\beta_{k+1}|}{\beta_{k+1}^{\mathbb{FR}}}\sum_{j=0}^{k-1}\sigma^j \le \frac{g_{k+1}^T d_{k+1}}{\|g_k\|^2} \le -1 + \sigma\frac{|\beta_{k+1}|}{\beta_{k+1}^{\mathbb{FR}}}\sum_{j=0}^{k-1}\sigma^j$$

. 由于 $|\beta_k| \leq \beta_k^{\mathbb{FR}}$，并且 $\sum_{j=0}^{k-1} \sigma^{j+1} = \sum_{j=0}^{k} \sigma^j - 1$, 我们可以得到定理**??**对于 $k+1$ 也成立，故数学归纳法成立。 □

如果采用非精确线搜索，FR 和 PRP 方法都有可能产生式上升的方向。而该定理告诉我们对于 FR 方法而言，只有当使用强 Wolfe 线搜索，并且保证 $\sigma \in (0, \frac{1}{2})$ 时，得到的方向才是下降方向。

定理 8.3 (使用精确线搜索的 FR 方法的收敛性). [1992Global] 设有 $f(x)$ 下界，$g(x)$ 满足 Lipschitz 条件，对使用精确线搜索准则的 FR 方法，则存在 N，使得 $g_N = 0$，或者 $\lim_{k\to\infty} inf\|g_k\| = 0$

证明: 根据定理**??**以及在强 Wolfe 准则下的 FR 公式 $|g(x_k + \alpha_k d_k)^T d_k| \leq -\sigma g_k^T d_k$, 我们得到

$$
\begin{aligned}
|g_k^T d_{k-1}| &\leq -\sigma g_{k-1}^T d_{k-1} \\
&\leq \sigma \sum_{j=0}^{k-2} \sigma^j \|g_{k-1}\|^2 \leq \frac{\sigma}{1-\sigma} \sigma^j \|g_{k-1}\|^2.
\end{aligned}
$$

根据梯度下降方向以及定理**??**，可以得到

$$
\begin{aligned}
\|d_k\|^2 &\leq \|g_k\|^2 + 2|\beta_k| g_k^T d_{k-1} + \beta_k^2 \|d_{k-1}\|^2 \\
&\leq \|g_k\|^2 + \frac{2\sigma}{1-\sigma} |\beta_k| \|g_{k-1}\|^2 + \beta_k^2 \|d_{k-1}\|^2 \leq \frac{1+\sigma}{1-\sigma} \|g_k\|^2 + \beta_k^2 \|d_{k-1}\|^2.
\end{aligned}
$$

重复使用这一关系，我们定义 $\hat{\sigma} := \frac{1+\sigma}{1-\sigma} \geq 1$，并且使用条件 $|\beta_k| \leq \beta_k^{\mathbb{FR}}$，得到

$$
\begin{aligned}
\|d_k\|^2 &\leq \hat{\sigma} \|g_k\|^2 + \beta_k^2 (\hat{\sigma} \|g_{k-1}\|^2 + \beta_{k-1}^2 \|d_{k-2}\|^2) \\
&\leq \hat{\sigma} \|g_k\|^4 + \sum_{j=1}^{k} \|g_j\|^{-2}.
\end{aligned}
$$

我们假设对于所有的 k，都有 $\|g_k\| \geq \gamma > 0$，这表示

$$
\|d_k\|^2 \leq \frac{\hat{\sigma} \gamma^4}{\gamma^2} k. \tag{8.6}
$$

由此得证 □

该定理从理论上证明了精确线搜索下 FR 方法的收敛性，通过定理**??**，可以直到使用强 Wolfe 线搜索的 FR 方法在 $\sigma \in (0, \frac{1}{2})$ 时有同样的收敛结果。而证明这一定理我们使用了

Zoutendijk 条件，其使用率相对广泛。

定理 8.4 (Zoutendijk 条件). 设有 $f(x)$ 下界，$g(x)$ 满足 Lipschitz 条件，使用 Wolfe 线搜索或精确线搜索准则的，具有 $x_{k+1} = x_k + \alpha_k d_k$ 迭代格式的一般下降方法满足 Zoutendijk 条件：

$$\sum_{k\geq 0} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty \tag{8.7}$$

证明: 设存在常数 η, 对任意 $k \geq 0$，有. 利用 Zoutendijk 条件

$$\sum_{k\geq 0} \frac{(g_k^{\mathbb{T}} d_k)^2}{\|d_k\|^2} < \infty = \sum_{k\geq 0} \|g_k\|^2 \cos^2\theta_k < \infty,$$

, 其中 $\theta_k =< d_k, -g_k >$, 知 $\lim\limits_{k\to\infty} \cos\theta_k = 0$.

考虑到FR方法的迭代格式和精确线搜索的特点，在第 k 次迭代，我们有 $\|d_k\| = \|g_k\| \sec\theta_k$；在第 $k+1$ 次迭代，我们有 $\beta_k\|d_k\| = \|g_{k+1}\| \tan\theta_{k+1}$. 结合这两个式子以及 β_k 的表示，就有

$$\tan\theta_{k+1} = \sec\theta_k \frac{\|g_{k+1}\|}{\|g_k\|}$$

, 上式两边平方，并利用 $\sec^2\theta_k = 1 + \tan^2\theta_k$，便得到

$$\frac{\tan^2\theta_{k+1}}{\|g_{k+1}\|^2} = \frac{1}{\|g_k\|^2} + \frac{\tan^2\theta_k}{\|g_k\|^2}$$

. 利用这个关系递推，可以得到

$$\frac{\tan^2\theta_k}{\|g_k\|^2} = \sum_{i=0}^{k-1} \frac{1}{\|g_i\|^2}$$

, 其中 $d_0 = -g_0$，从而

$$\frac{\tan^2\theta_k}{\|g_k\|^2} \leq \frac{k}{\eta^2}$$

, 即

$$\frac{\eta^2}{k} \leq \frac{\|g_k\|^2}{\tan^2\theta_k} = \|g_k\|^2 \cot^2\theta_k$$

. 当充分大时，因为 $\cot^2\theta_k \leq 2\cos^2\theta_k$，就有

$$\sum_{k\geq 0} \|g_k\|^2 \cot^2\theta_k = \infty$$

. 这与 Zoutendijk 条件矛盾。 □

9. 增广拉格朗日法与交替方向乘子法

本节引入增广拉格朗日函数的概念，分析增广拉格朗日法[1976AugLagrange] 与交替方向乘子法[2017ADMM]，并使用两种方法求解具体的凸优化问题。

9.1 凸优化模型

考虑带约束条件的凸优化问题

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \\ &\text{s.t.} \qquad \boldsymbol{Ax}=\boldsymbol{b} \end{aligned} \tag{9.1}$$

其中，$f(x)$ 是一个连续凸函数，这样的问题一般可以由增广拉格朗日法求解。

而对于如下的约束凸优化问题

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) := f_1(\boldsymbol{x})+f_2(\boldsymbol{x}) \\ &\text{s.t.} \qquad \boldsymbol{A_1x_1}+\boldsymbol{A_2x_2}=\boldsymbol{b} \end{aligned} \tag{9.2}$$

其中，$f_1(x), f_2(x)$ 都是适当的闭凸函数，同时 $f(x)$ 可以被分解成两个分离的部分 $f_1(x)$ 和 $f_2(x)$，但是这两部分被线性约束结合在一起，这样的问题可以由交替方向乘子法求解。通过对该问题应用交替方向乘子法求解，可以分别更新 x_1 和 x_2 的参数，提高了并行计算的效率。

9.2 增广拉格朗日函数

在拉格朗日函数的基础上，增加二次罚函数，可以得到增广拉格朗日函数

$$L_\beta(x,\lambda)=f(x)+\sum_{i=1}^{m}\lambda_i c_i(x)+\frac{\beta}{2}\sum_{i=1}^{m}c_i^2(x), \tag{9.3}$$

其中，式中的 β 称为罚因子，$c_i(x), i=1,2,\cdots,m$ 表示惩罚项。

特别地，当约束关系是线性的时候，即上式中的惩罚项 $c(\boldsymbol{x})=\boldsymbol{Ax}-\boldsymbol{b}$ 时，可以得到增广拉格朗日函数

$$L_\beta(\boldsymbol{x},\lambda)=f(\boldsymbol{x})-\lambda^T(\boldsymbol{Ax}-\boldsymbol{b})+\frac{\beta}{2}\|\boldsymbol{Ax}-\boldsymbol{b}\|_2^2. \tag{9.4}$$

考虑增广拉格朗日函数**??**，使 $L_\beta(\boldsymbol{x},\lambda)$ 值最小的点 x^{k+1} 满足

$$\nabla_x L_\beta(x^{k+1},\lambda^k)=\nabla f(x^{k+1})+\sum_{i=1}^{m}(\lambda_i^k+\beta c_i(x^{k+1}))\nabla c_i(x^{k+1})=0.$$

因此，根据上式可以得到，对于凸优化问题**??**，存在最优解 x^* 与最优乘子 λ^*，有

$$\nabla f(x^*)+\sum_{i=1}^{m}\lambda_i^*\nabla c_i(x^*)=0. \tag{9.5}$$

9.3 增广拉格朗日法

对于凸优化问题**??**即

$$\begin{aligned}&\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x})\\ &\text{s.t.}\qquad \boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}\end{aligned}$$

使用一般的增广拉格朗日函数**??**，增广拉格朗日法有如下的迭代格式

$$\begin{cases}\boldsymbol{x}^{k+1} &= \underset{\boldsymbol{x}\in\mathbb{R}^n}{\operatorname{argmin}} L_{\beta_k}(\boldsymbol{x}^k,\lambda^k)\\ \lambda^{k+1} &= \lambda^k+\beta_k c(x^{k+1})\end{cases} \tag{9.6}$$

其中，$\boldsymbol{x}^{k+1}$ 表示第 $k+1$ 次迭代的结果，同样地，β_k 表示第 k 次迭代的罚因子，参数的值可以在每次迭代中进行更新。

结合迭代公式**??**与初始化设置，可以将增广拉格朗日法总结为如下算法。

算法 10 增广拉格朗日法

输入: $f(x)$ 与其增广拉格朗日函数 $L_{\beta_k}(x^k,\lambda^k)$，初始化设置 $k=0$，并且给定罚因子的更新常数 $\rho>0$

输出: 最优解 x^* 与最优乘子 λ^*

while 未收敛 **do**

$\boldsymbol{x}^{k+1}=\underset{x\in\mathbb{R}^n}{\operatorname{argmin}} L_{\beta_k}(x^k,\lambda^k)$

$\lambda^{k+1}=\lambda^k+\beta_k c(x^{k+1})$

$\beta_{k+1}=\rho\beta_k$

end

9.4 交替方向乘子法

对于凸优化问题**??**即

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) := f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}) \\ &\text{s.t.} \qquad \boldsymbol{A_1 x_1} + \boldsymbol{A_2 x_2} = \boldsymbol{b} \end{aligned}$$

由于其目标函数可以被两个线性约束分离，因此适用于交替方向乘子法求解。

首先，注意到这里的约束条件都是线性的，因此由增广拉格朗日函数**??**，可以得到

$$L_\beta(\boldsymbol{x_1}, \boldsymbol{x_2}, \lambda) = f_1(\boldsymbol{x_1}) + f_2(\boldsymbol{x_2}) - \lambda^T(\boldsymbol{A_1 x_1} + \boldsymbol{A_2 x_2} - \boldsymbol{b} + \frac{\beta}{2}\|\boldsymbol{A_1 x_1} + \boldsymbol{A_2 x_2} - \boldsymbol{b}\|_2^2). \tag{9.7}$$

由上式可以进一步使用增广拉格朗日法得到迭代格式

$$\begin{cases} \boldsymbol{x_1^{k+1}}, \boldsymbol{x_2^{k+1}} &= \underset{\boldsymbol{x_1}, \boldsymbol{x_2}}{\operatorname{argmin}} L_\beta(\boldsymbol{x_1^k}, \boldsymbol{x_2^k}, \lambda^k) \\ \lambda^{k+1} &= \lambda^k - \beta(\boldsymbol{A_1 x_1^{k+1}} + \boldsymbol{A_2 x_2^{k+1}} - \boldsymbol{b}) \end{cases} \tag{9.8}$$

注意到上式对于变量 $\boldsymbol{x_1}, \boldsymbol{x_2}$ 的迭代格式与增广拉格朗日法不同的地方在于，上式同时更新了两个变量 $\boldsymbol{x_1}$ 和 $\boldsymbol{x_2}$，这对于计算求解而言增加了复杂度，因此交替方向乘子法将两个变量分开求解，这样可以充分提升计算效率，下面给出交替方向乘子法的迭代格式

$$\begin{cases} \boldsymbol{x_1^{k+1}} &= \underset{\boldsymbol{x_1}}{\operatorname{argmin}} L_\beta(\boldsymbol{x_1^k}, \boldsymbol{x_2^k}, \lambda^k) \\ \boldsymbol{x_2^{k+1}} &= \underset{\boldsymbol{x_2}}{\operatorname{argmin}} L_\beta(\boldsymbol{x_1^{k+1}}, \boldsymbol{x_2^k}, \lambda^k) \\ \lambda^{k+1} &= \lambda^k - \beta(\boldsymbol{A_1 x_1^{k+1}} + \boldsymbol{A_2 x_2^{k+1}} - \boldsymbol{b}) \end{cases} \tag{9.9}$$

通过交替求解两个变量 $\boldsymbol{x_1}$ 和 $\boldsymbol{x_2}$ 的极小，可以交替对其进行更新，并在完成两个变量的更新后，再利用更新后的值对 λ 进行更新。

结合迭代公式**??**与初始化设置，可以将交替方向乘子法**2017ADMM** 总结为如下算法

9.5 应用

以 Lasso 问题为例，使用交替方向乘子法求解该问题。

算法 11 交替方向乘子法

输入: $f(x) := f_1(x) + f_2(x)$ 与其增广拉格朗日函数 $L_{\beta_k}(\boldsymbol{x_1^k}, \boldsymbol{x_2^k}, \lambda^k)$，初始化设置 $k = 0$，并且给定罚因子的更新常数 $\rho > 0$

输出: 最优解 $\boldsymbol{x_1^*}, \boldsymbol{x_2^*}$ 与最优乘子 λ^*

while 未收敛 **do**

$$\boldsymbol{x_1^{k+1}} = \underset{\boldsymbol{x_1}}{\operatorname{argmin}} L_\beta(\boldsymbol{x_1^k}, \boldsymbol{x_2^k}, \lambda^k)$$

$$\boldsymbol{x_2^{k+1}} = \underset{\boldsymbol{x_2}}{\operatorname{argmin}} L_\beta(\boldsymbol{x_1^{k+1}}, \boldsymbol{x_2^k}, \lambda^k)$$

$$\lambda^{k+1} = \lambda^k - \rho\beta(\boldsymbol{A_1 x_1^{k+1}} + \boldsymbol{A_2 x_2^{k+1}} - \boldsymbol{b})$$

end

问题 9.1. 求解 Lasso 问题，即求

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) := \frac{1}{2}\|\boldsymbol{Ax} - \boldsymbol{b}\|_2^2 + \mu\|\boldsymbol{x}\|_1. \tag{9.10}$$

解: 首先，引入变量 $\boldsymbol{x_1}$ 和变量 $\boldsymbol{x_2}$，使该问题转化成求解

$$\begin{aligned} &\min_{\boldsymbol{x_1}, \boldsymbol{x_2} \in \mathbb{R}^n} f(\boldsymbol{x}) := \frac{1}{2}\|\boldsymbol{Ax_1} - \boldsymbol{b}\|_2^2 + \mu\|\boldsymbol{x_2}\|_1 \\ &\text{s.t.} \boldsymbol{x_1} = \boldsymbol{x_2} \end{aligned} \tag{9.11}$$

这样，由线性约束关系可以将目标函数 $f(\boldsymbol{x})$ 分解为 $f_1(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{Ax_1} - \boldsymbol{b}\|_2^2$ 和 $f_2(\boldsymbol{x}) = \mu\|\boldsymbol{x_2}\|_1$

于是，可以由增广拉格朗日函数**??**得到，

$$L_\beta(\boldsymbol{x_1}, \boldsymbol{x_2}, \lambda) = \frac{1}{2}\|\boldsymbol{Ax_1} - \boldsymbol{b}\|_2^2 + \mu\|\boldsymbol{x_2}\|_1 - \lambda^T(\boldsymbol{x_1} - \boldsymbol{x_2}) + \frac{\beta}{2}\|\boldsymbol{x_1} - \boldsymbol{x_2}\|_2^2.$$

进一步由迭代公式**??**可以得到

$$\begin{cases} \boldsymbol{x_1^{k+1}} &= \underset{\boldsymbol{x_1}}{\operatorname{argmin}}\{\frac{1}{2}\|\boldsymbol{Ax_1^k} - \boldsymbol{b}\|_2^2 + \frac{\beta}{2}\|\boldsymbol{x^k} - \boldsymbol{x_2^k} - \frac{\lambda^k}{\beta}\|_2^2\} \\ &= (\boldsymbol{A^T A} + \beta\boldsymbol{I})^{-1}(\boldsymbol{A^T b} + \beta\boldsymbol{x_2^k} - \lambda^k) \\ \boldsymbol{x_2^{k+1}} &= \underset{\boldsymbol{x_2}}{\operatorname{argmin}}\{\mu\|\boldsymbol{x_2^k}\|_1 + \frac{\beta}{2}\|\boldsymbol{x_2^k} - \boldsymbol{x_1^{k+1}} + \frac{\lambda^k}{\beta}\|_2^2\} \\ &= \operatorname{prox}_{\frac{\mu}{\beta}\|\cdot\|_1}(\boldsymbol{x_1^{k+1}} + \frac{\lambda^k}{\beta}) \\ \lambda^{k+1} &= \lambda^k + \beta(\boldsymbol{x_1^{k+1}} - \boldsymbol{x_2^{k+1}}) \end{cases} \tag{9.12}$$

使用 Python 对该问题求解，可以得到结果如图**??**所示，代码在附录中给出。其中，beta, lambda 和 rho 分别同算法**??**中的 β, λ, ρ。可以看出当 $\lambda = 1$ 时，不论 β 和 ρ 取值如何，都能表现出较好的收敛效果；当 β 取值较大时，收敛速度显著提高。

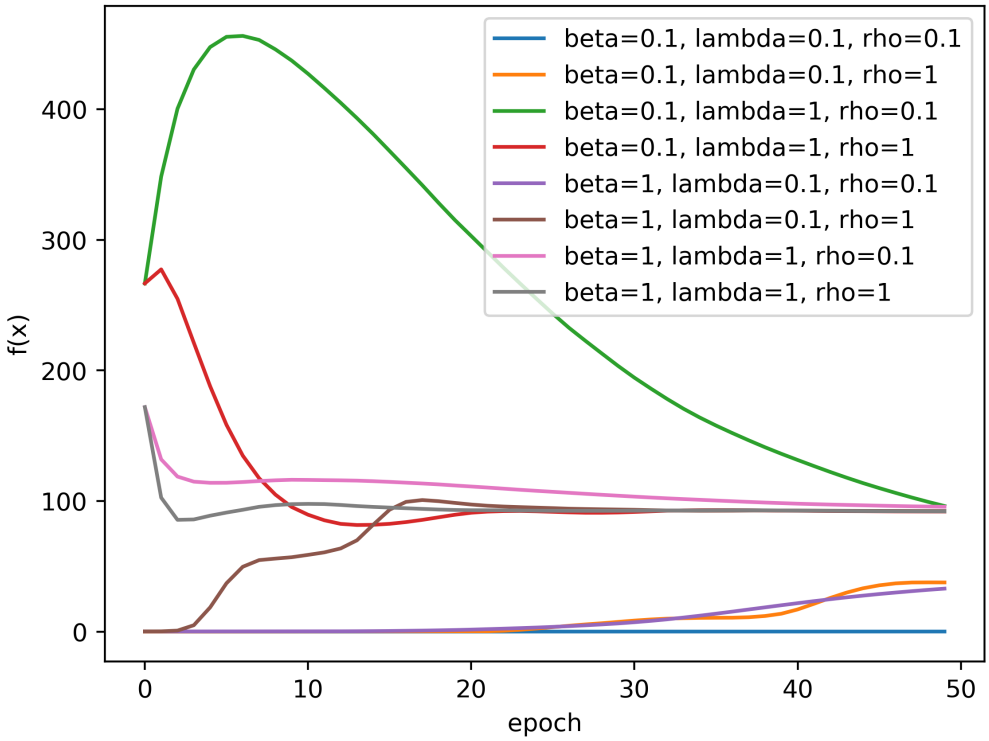

图 9-1: Python 实现交替方向乘子法求解 Lasso 问题

10. 总结

本文对于最优化问题分析并总结了多种求解方法，包括在无约束条件下的梯度下降法，随机梯度法，临近梯度法，牛顿迭代法，拟牛顿迭代法，共轭梯度法等；以及在有约束条件下的增广拉格朗日乘子法与交替方向乘子法。以下分别概括几类最优化方法的特点：

1) 梯度下降法

梯度下降法是一类非常基础的方法，其基本思想是根据步长和当前位置计算梯度，到达下一个迭代点。由于每次学习都使用整个函数，所以最终能保证收敛于极值点，凸函数收敛于全局极值点，非凸函数可能收敛于局部极值点，而全局的使用导致这种方法学习时间过长，消耗资源。

2) 随机梯度法

随机梯度下降是梯度下降法的一种求解思路，其可以被用于解决梯度下降法的弊端，在随机梯度下降法中，每次迭代可以只用一个训练数据来更新参数。相较于一般梯度下降法，其训练速度快，但代价是准确度下降、不是全局最优、不易于并行实现随机梯度下降会最小化所有训练样本的损失函数，使得最终解为全局最优解，即求解参数会使得风险函数最小；虽不是每次迭代得到的损失函数都向着全局最优方向，但整体方向是全局最优解，且最终结果往往在全局最优解附近。

3) 临近梯度法

临近梯度法是一种特殊的梯度下降方法，主要用于求解目标函数不可微的最优化问题。如果目标函数在某些点是不可微的，那么该点的梯度无法求解，传统的梯度下降法也就无法使用。该算法的思想是，使用临近算子作为近似梯度，进行梯度下降。其收敛性结果同梯度下降法一致，但是收敛速度在实际中略小于梯度下降法

4) 牛顿迭代法

牛顿法收敛很快，对于二次函数只迭代一次便达到最优点，对于二次函数也能较快迭代到最优点，但要计算二阶偏导数矩阵及其逆矩阵，对维数较高的优化问题，其计算工作和存储量都太大。

5) 拟牛顿迭代法

拟牛顿法的本质思想是改善牛顿法每次需要求解复杂的 Hesse 矩阵的逆矩阵的缺陷，它使用正定矩阵来近似 Hesse 矩阵的逆，从而简化了运算的复杂度；而且有时候目标函数的 Hesse 矩阵无法保证正定。拟牛顿法只要求每一步迭代时知道目标函数的梯度，通过测量梯度的变化，构造一个目标函数的模型使之足以产生超线性收敛性。另外，因为拟牛顿

法不需要二阶导数的信息，所以有时比牛顿法更为有效。如今，优化软件中包含了大量的拟牛顿算法用来解决无约束，约束，和大规模的优化问题。常用的拟牛顿法有 DFP 算法和 BFGS 算法。

6) 共轭梯度法

共轭梯度法是介于梯度下降法与牛顿法之间的一个方法，它仅需利用一阶导数信息，但克服了梯度下降法收敛慢的缺点，又避免了牛顿法需要存储和计算 Hesse 矩阵并求逆的缺点，共轭梯度法不仅是解决大型线性方程组最有用的方法之一，也是解大型非线性最优化最有效的算法之一。在各种优化算法中，共轭梯度法是非常重要的一种。其优点是所需存储量小，具有步收敛性，稳定性高，而且不需要任何外来参数。

7) 增广拉格朗日乘子法

增广拉格朗日方法在传统拉格朗日方法的基础上，针对函数对应的拉格朗日函数添加罚项。只要函数的约束条件不满足，就会根据它脱轨的情况来做一定的惩罚，这样做可以把一个带约束优化的问题重新变回了无约束优化问题。

8) 交替方向乘子法

交替方向乘子法是为了解决增广拉格朗日算法不能做分解的问题。它考虑了自变量由两个部分组合而成时候如何分解优化的问题。其大致思想是是交替地根据某一个变量求最值，同时固定住其余变量。如此可以把原本可能不凸问题变成凸问题。

A 附录

1.1 环境要求

Python >= 3

Numpy == 1.18.1

Matplotlib == 3.1.3

1.2 梯度下降法求解最小二乘问题

```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(1234)

len_x = 1000
A = np.random.rand(len_x, len_x)
x_star = np.random.rand(len_x, 1)
b = A*x_star
my_x = np.random.rand(len_x, 1)

def get_grad(x):
    return 2*A.T*(A*x-b)

def get_loss(x):
    return np.linalg.norm(A*x-b, ord=1)**2

delta = 1e10
alpha = 5e-1
iter = 0
iter_list = []
delta_list = []
# while delta > 1e-4:
# 手动设置迭代1000次
while iter<1e3:
```

```
    new_x = my_x - alpha*get_grad(my_x)
    delta = np.linalg.norm(new_x - x_star, ord=2) / np.linalg.norm(x_star, ord=2)
    iter_list.append(iter)
    delta_list.append(np.log(delta))
    print(f'{iter}, loss is {get_loss(my_x)}, rela_x is {delta}')
    my_x = new_x
    iter += 1

plt.plot(iter_list, delta_list)
plt.xlabel('epoch')
plt.ylabel('log ||newx-x||')
plt.savefig('gd-loss.png', dpi=400)
```

1.3 临近梯度下降法求解 Lasso 问题

```
import numpy as np
import matplotlib.pyplot as plt

def generate_data(m=1000, n=100, density=0.4):
    "Generates data matrix X and observations Y."
    np.random.seed(1)
    x_star = np.random.random((m,1))
    idxs = np.random.choice(range(m), int((1-density)*m), replace=False)
    for idx in idxs:
        x_star[idx] = 0
    A = np.random.random((n, m))
    b = A.dot(x_star)
    return x_star, A, b


m, n, density = 1000, 100, 0.2
mu = 1
```

```
x_star, A, b = generate_data(m, n, density)
my_x = np.random.random((m, 1))


def f(A, x1, x2, b, mu):
    """
    对于临近梯度下降法，x1=x2，对于ADMM，将目标问题拆分为两部分
    """
    return 0.5 * np.linalg.norm(np.dot(A, x1) - b, ord=2) + mu * np.linalg.norm(x2,
        ord=1)


def get_gradient(A, x, b):
    return np.dot(A.transpose(), np.dot(A, x) - b)


def get_prox(learning_rate, x, mu):
    maxPart = np.abs(x) - learning_rate * mu

    for i in range(len(maxPart)):
        maxPart[i] = 0 if maxPart[i] <= 0 else maxPart[i]

    return np.sign(x) * maxPart


def step_proximal(A, x, b, mu, learning_rate):
    return get_prox(learning_rate, x - learning_rate * get_gradient(A, x, b), mu)


def step_admm(A, x1, x2, b, mu, beta, lamb, rho):
    size = np.shape(A)[1]
    newX1 = np.dot(np.linalg.inv(np.dot(A.transpose(), A) + beta *
```

```
        np.identity(size)), np.dot(A.transpose(), b) + beta * x2 - lamb)
    newX2 = get_prox(mu/beta, newX1+1/beta*lamb, mu)
    newLamb = lamb + rho * beta * (newX1 - newX2)

    return newX1, newX2, newLamb

def proximal_gradient_descent(A, x, b, mu, learning_rate):
    delta = 1e10
    epoch = 0
    iterationList = []
    deltaList = []

    while delta > 1e-04 and epoch < 300:
        newX = step_proximal(A, x, b, mu, learning_rate)
        delta = np.abs(f(A, newX, newX, b, mu))
        iterationList.append(epoch)
        deltaList.append(delta)
        x = newX
        epoch += 1

    curve, = plt.plot(iterationList, deltaList, '-')
    return curve
```

1.4 交替方向乘子法求解 Lasso 问题

```
def admm(A, x, b, mu, beta, lamb, rho):
    x1 = x
    x2 = x
    lamb = lamb * np.ones((np.shape(A)[1], 1))
    delta = 1e10
    epoch = 0
    iterationList = []
```

```
deltaList = []

while delta > 1e-04 and epoch < 50:
    newX1, newX2, newLamb = step_admm(A, x1, x2, b, mu, beta, lamb, rho)
    delta = np.abs(f(A, newX1, newX2, b, mu))
    iterationList.append(epoch)
    deltaList.append(delta)
    x1 = newX1
    x2 = newX2
    lamb = newLamb
    epoch += 1

curve, = plt.plot(iterationList, deltaList, '-')
return curve
```