中国科学 : 数学　2020 年　第 50 卷　第 9 期 : 1271 ∼ 1336
SCIENTIA SINICA Mathematica
论　文

优化算法的复杂度分析

献给越民义教授 100 华诞

王奇超[1]，文再文[1*]，蓝光辉[2]，袁亚湘[3]

1. 北京大学北京国际数学研究中心, 北京 100871;
2. H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA;
3. 中国科学院数学与系统科学研究院计算数学与科学工程计算研究所, 北京 100190
E-mail: chichauwang@gmail.com, wenzw@pku.edu.cn, george.lan@isye.gatech.edu, yyx@lsec.cc.ac.cn

收稿日期: 2018-10-17; 接受日期: 2019-07-17; 网络出版日期: 2020-09-11; * 通信作者
国家自然科学基金 (批准号: 11831002 和 11331012) 资助项目

摘要　优化算法的收敛性分析是优化中很重要的一个领域, 然而收敛性并不足以作为比较不同算法效率的标准, 因此需要另外一套衡量优化问题难易程度以及优化算法效率高低的理论, 这套理论被称为优化算法的复杂度分析理论. 本文共分为 5 个部分. 第 1 节介绍复杂度分析的背景和理论框架, 给出复杂度分析的定义、方法和例子, 并总结本文中的复杂度结论. 第 2 节介绍光滑优化问题的复杂度分析, 给出不同优化问题的复杂度上界和下界, 并给出加速梯度法收敛性分析的框架. 第 3 节介绍非光滑优化问题的复杂度上界, 介绍次梯度法、重心法、椭球法和近似点梯度法的复杂度分析. 第 4 节介绍条件梯度法的复杂度分析, 介绍条件梯度法的复杂度上界和下界, 以及加速条件梯度法的框架. 第 5 节介绍随机优化算法的复杂度分析, 比较随机优化算法在凸和非凸问题下收敛的置信水平和复杂度.

关键词　优化算法　复杂度分析　加速梯度法　条件梯度法　随机优化算法

MSC (2010) 主题分类　65K05, 65Y20, 90C15, 90C60

1　复杂度分析的理论基础

1.1　引言

优化算法的复杂度分析一直以来是我们理解优化算法和设计新的优化算法的工具. 考虑通常形式下的优化问题, 记作 $\mathcal{P}$:

$$f^* = \min_{x \in X} f(x). \tag{1.1}$$

根据约束集合 X 类型和目标函数 $f(x)$ 类型对上述优化问题进行分类:

英文引用格式: Wang Q C, Wen Z W, Lan G H, et al. Complexity analysis for optimization methods (in Chinese). Sci Sin Math, 2020, 50: 1271–1336, doi: 10.1360/N012018-00251

- 约束/无约束优化问题: $X \subset \mathbb{R}^n / X \equiv \mathbb{R}^n$;
- 光滑/非光滑优化问题: $f(x)$ 在 X 上可/不可导;
- 凸/强凸/非凸优化问题: $f(x)$ 是凸/强凸/非凸函数;
- 随机优化问题: $f(x) = \mathbb{E}_\xi[F(x,\xi)]$, 其中 ξ 是一个随机变量, $\mathbb{E}_\xi[F(x,\xi)]$ 表示 $F(x,\xi)$ 的期望.

为了分析优化算法在求解问题 (1.1) 时的效率, 我们将引入优化算法的复杂度分析的概念.

将 (1.1) 的最优解和最优值分别记为 x^* 和 f^*. 我们知道一旦给定 $f(x)$ 和约束集合 X 的具体形式, 我们可以找到不同的数值算法, 记作 $\mathcal{S}$, 来求得 x^* 和 f^* 给定误差 ϵ 的近似值. 为了寻找更好的求解算法, 我们需要衡量不同算法 $\mathcal{S}$ 的效率. 而复杂度分析理论就提供了一套衡量算法效率的框架.

当 (1.1) 中 $f(x)$ 和 X 具体形式给定时, 我们将 (1.1) 称为问题 $\mathcal{P}$. 把具备某种共同性质的问题 $\mathcal{P}$ 集合称为问题集合 $\mathcal{F}$. 定义问题 $\mathcal{P}$ 的解算法集合 $\mathcal{M} := \mathcal{M}(\mathcal{P}, \epsilon)$ (solution methods set), 其中包含所有可以求解 $\mathcal{P}$ 到某一给定误差精度 ϵ 的解算法 (solution method), 记作 $\mathcal{S} := \mathcal{S}(\mathcal{P}, \epsilon) \in \mathcal{M}$. 一个自然的问题是, 在 $\mathcal{P}$ 的解算法集合 $\mathcal{M}$ 中是否存在效率最好的解算法 $\mathcal{S}$?

当然, 我们还没有正式给出衡量算法效率的度量. 由于无法完全穷尽 $\mathcal{M}$ 的集合, 即使我们设定了算法效率的度量, 也很难找到效率最好的算法. 事实上, 考虑这样一个求解问题 $\mathcal{P}$ 的解算法 $\mathcal{S}_0$. 当调用 $\mathcal{S}_0$ 时, 它只执行一个操作: 返回 $x^* = 0$. 对于最优解不是 $x^* = 0$ 的某些问题 $\mathcal{P}$, $\mathcal{S}_0$ 不是它的解算法, 但是当 $\mathcal{P}$ 的最优解刚好是 $x^* = 0$ 时, 对于该问题 $\mathcal{P}$ 来说, $\mathcal{S}_0$ 在其解算法集合 $\mathcal{M}$ 中的效率是无法超越的. 然而 $\mathcal{S}_0$ 这样的解算法并不是我们寻找的效率最好的解算法.

因此, 我们无法寻找求解某一特定问题 $\mathcal{P}$ 时效率最高的解算法, 但是我们可以寻找求解某一类问题集合 $\mathcal{F} \ni \mathcal{P}$ 时效率最好的解算法. 事实上, 设计数值优化算法的目的都是想用它求解某一类具有相同特征的问题. 因此, 我们需要定义解算法 $\mathcal{S}$ 在一类问题集合 $\mathcal{F}$ 上的表现来衡量一个解算法的效率.

本节介绍优化算法复杂度分析的理论基础. 复杂度分析的理论分为三个部分, 第一部分是对所要求解的优化问题进行分类, 建立问题模型; 第二部分是对求解优化问题的数值算法进行抽象, 建立算法模型; 第三部分是定义算法模型在问题模型上的度量, 即复杂度. 本节先分别介绍复杂度分析的三个部分, 最后用两个具体的例子介绍复杂度分析的分析方法.

1.2 复杂度分析的问题模型

复杂度分析的问题模型分为三个部分: 全局信息、局部信息和解的精度. 问题模型是对问题集合 $\mathcal{F}$ 更精确的描述, 也是对所要求解的优化任务更具体的要求.

当考察数值解算法 $\mathcal{S}$ 在优化问题集合 $\mathcal{F}$ 上的表现时, 假设算法 $\mathcal{S}$ 无法获得所要求解的具体问题 $\mathcal{P}$ ($\mathcal{P} \in \mathcal{F}$) 的全部信息, 只能获取 $\mathcal{P}$ 在 $\mathcal{F}$ 中的共有特征信息, 例如, 问题集合 $\mathcal{F}$ 中的目标函数是否光滑, 可微, 其约束集合的类型是否有界等. 我们将 $\mathcal{F}$ 中所具有的共有特征信息记为 Σ, 表示问题集合 $\mathcal{F}$ 的全局信息.

为了认识和求解 $\mathcal{F}$ 中的某一具体问题 $\mathcal{P}$, $\mathcal{S}$ 需要逐步去收集有关 $\mathcal{P}$ 的局部信息, 获得 $\mathcal{P}$ 的局部几何结构, 然后根据收集的历史局部信息给出寻找最优解的策略. 我们将解算法 $\mathcal{S}$ 收集局部信息数据的过程称记作子程序 (oracle) $\mathcal{O}$. 子程序 $\mathcal{O}$ 是一个单元, 算法 $\mathcal{S}$ 可以连续调用它来获得一系列有关 $\mathcal{P}$ 的局部信息. 一个具体的子程序例子是梯度法中求解函数导数的过程, 导数反应了目标函数的局部几何信息, 梯度法的执行过程需要连续地调用有关求解目标函数导数的子程序.

为了衡量算法的效率, 我们需要事先给出数值算法解的精度的信息, 记作 $\mathcal{T}_\epsilon$. 不同的问题集合 $\mathcal{F}$

接受不同类型解的精度的度量方式. 我们也将解的精度信息固定到问题集合 $\mathcal{F}$ 里面, 作为算法求解 $\mathcal{F}$ 时可调用的信息.

因此, 我们扩充问题集合 $\mathcal{F}$ 得到复杂度分析理论中的问题模型 $\mathcal{F}$:

$$\mathcal{F} \equiv (\Sigma, \mathcal{O}, \mathcal{T}_\epsilon). \tag{1.2}$$

在解算法 $\mathcal{S}$ 求解问题集合 $\mathcal{F}$ 中的优化问题时, 它只能连续调用有关 $\mathcal{F}$ 的三部分信息: 全局信息 Σ、局部信息 $\mathcal{O}$ 和解的精度 $\mathcal{T}_\epsilon$, 并利用这些信息来获得最优解的近似值. 下面对问题模型的三部分信息做更细致的描述.

1.2.1 全局信息 Σ

全局信息由 $\mathcal{F}$ 中问题的目标函数信息和约束集合信息组成. 对于目标函数信息, 我们主要研究求解下列函数空间中的问题时的算法复杂度:

- $f(x)$ 是凸函数、强凸函数和非凸函数的情形, $f(x)$ 是光滑函数和非光滑函数的情形.
- $f(x)$ 是复合函数

$$f(x) = g(x) + h(x). \tag{1.3}$$

在机器学习问题中 $g(x)$ 光滑, 正则项 $h(x)$ 可能光滑也可能不光滑.

- $f(x)$ 是随机优化问题,

$$\min_{x\in X}\{f(x) := \mathbb{E}[F(x,\xi)]\}, \tag{1.4}$$

其中 ξ 是随机变量.

更具体地, 我们可以将凸函数空间做划分.

定义 1.1 (凸函数子集) 设 $X \subset \mathbb{R}^n$ 是闭凸集合, $f : X \to \mathbb{R}$ 是凸函数, X^* 是 (1.1) 最优解的集合, x_* 是 X 在最优解集合 X^* 上的投影. 定义如下 f 的凸函数子集:

(1) $C(X) = C^0(X)$ 是所有连续函数的集合.

(2) $C_L(X) = C_L^{0,0}(X)$: 若 $f(x) \in C_L(X)$, 则 $f(x)$ 具有 Lipschitz 连续性:

$$|f(x) - f(y)| \leqslant L\|x - y\|, \quad \forall\, x, y \in X. \tag{1.5}$$

(3) $C_L^{1,1}(X)$: 若 $f(x) \in C_L^{1,1}(X)$, 则 $f(x)$ 一阶可导且导数具有 Lipschitz 连续性:

$$|\nabla f(x) - \nabla f(y)| \leqslant L\|x - y\|, \quad \forall\, x, y \in X. \tag{1.6}$$

(4) $C_L^{1,\alpha}(X)$: 若 $f(x) \in C_L^{1,\alpha}(X)$, 则 $f(x)$ 一阶可导且导数具有 Hölder 连续性, 其中 $\alpha \in (0, 1]$,

$$|\nabla f(x) - \nabla f(y)| \leqslant L\|x - y\|^\alpha, \quad \forall\, x, y \in X. \tag{1.7}$$

(5) $\mathcal{F}_{L,\mu}^{0,1}(X)$ 是 $C_L(X)$ 中所有强凸函数组成的集合. 若 $f(x) \in \mathcal{F}_{L,\mu}^{0,1}(X)$, 则

$$f(y) \geqslant f(x) + \langle \partial f(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2, \quad \forall\, x, y \in X. \tag{1.8}$$

(6) $\mathcal{F}_{L,\mu}^{1,1}(X)$ 是 $C_L^{1,1}(X)$ 中所有强凸函数组成的集合. 若 $f(x) \in \mathcal{F}_{L,\mu}^{1,1}(X)$, 则

$$f(y) \geqslant f(x) + \langle \partial f(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2, \quad \forall\, x, y \in X. \tag{1.9}$$

(7) $\mathcal{W}_{L,\mu}^{1,1}(X)$ 是 $C_L^{1,1}(X)$ 中所有弱强凸函数组成的集合. 若 $f(x)\in\mathcal{W}_{L,\mu}^{1,1}(X)$, 则

$$f^*\geqslant f(x)+\langle\nabla f(x),x_*-x\rangle+\frac{\mu}{2}\|x-x_*\|^2. \tag{1.10}$$

(8) $\mathcal{S}_{L,\mu}^{1,1}(X)$: 若 $f(x)\in\mathcal{S}_{L,\mu}^{1,1}(X)$, 则 $f(x)\in C_L^{1,1}(X)$ 且具有二阶增长性:

$$f(x)-f^*\geqslant\frac{\mu}{2}\|x-x_*\|^2. \tag{1.11}$$

对于弱强凸函数集合 $\mathcal{W}_{L,\mu}^{1,1}(X)$ 和具有二阶增长性的函数集合 $\mathcal{S}_{L,\mu}^{1,1}(X)$, 我们会在后面介绍其具体定义. 若记 $\mathcal{F}_L^{1,1}(X)$ 表示 $C_L^{1,1}(X)$ 与凸函数集合的交集, 文献 [1] 给出了凸函数集合之间的包含关系.

定理 1.1 $\mathcal{F}_{L,\mu}^{1,1}(X)\subset\mathcal{W}_{L,\mu}^{1,1}(X)\subset\mathcal{S}_{L,\mu}^{1,1}(X)\subset\mathcal{F}_L^{1,1}(X)$.

对于约束集合信息, 我们可以假定 X 是凸集且是闭的, 可以在 X 上做投影, 用投影梯度法 (projected gradient method) 求解 (1.1). 另一种情形下, 当 $X=\{x\in\mathbb{R}^n:\sum_{i=1}^n x_n=1,\ x_i\geqslant 0,\ i=1,\dots,n\}$ 是单纯形状或凸多面体时, 我们可以使用条件梯度法 (conditional gradient method) 求解 (1.1).

1.2.2 局部信息 $\mathcal{O}$

在复杂度分析中, 算法通过子程序来获得所求问题的局部信息. 不同的算法需要获得不同的局部信息. 例如, 对于梯度法, 任意给定输入 $x_0\in X$, 子程序返回函数值信息 $f(x_0)$ 和梯度信息 $\nabla f(x_0)$; 对于次梯度法, 则需要返回次梯度信息 $\partial f(x_0)$; 对于 Newton 法, 需要返回二阶导数信息 $\nabla^2 f(x_0)$.

这些子程序都是事先给定的, 复杂度分析理论并不衡量这些子程序的求解复杂度. 假设这些子程序 $\mathcal{O}$ 具有局部性和黑盒性.

(1) 子程序 $\mathcal{O}$ 是唯一局部信息来源: 数值算法唯一可获得的有关目标优化问题的局部信息来源于子程序;

(2) 子程序 $\mathcal{O}$ 具有局部稳定性: 调用子程序时, 对测试点 x 做微小扰动, 返回的局部信息 $\mathcal{O}(x)$ 变化不大,

其中第二条是对算法进行收敛性分析的关键假设. 例如, 在分析梯度法的收敛性时, 由于 $\mathcal{O}(x)=f(x)$ 或 $\mathcal{O}(x)=\nabla f(x)$, 我们需要假设存在某一常数 L 使得

$$\|\mathcal{O}(x)-\mathcal{O}(y)\|\leqslant L\|x-y\|, \tag{1.12}$$

也就是假设目标函数具有 Lipschitz 连续性, 或者目标函数的导数具有 Lipschitz 连续性. 对于没有这样性质的优化问题的目标函数, 数值算法有可能很难收敛, 或者收敛性质很差.

对于不同的算法, 我们总能抽象出该算法所需要的子程序, 大致分为如下类别:

- $\mathcal{ZO}$ (zero order oracle) : 适用于无导数优化算法, 对于任意给定的 x_0 返回函数值 $f(x_0)$.
- $\mathcal{FO}$ (first order oracle): 适用于梯度法, 返回函数在 x_0 处的函数值和一阶导数信息, $f(x_0)$, $\nabla f(x_0)$ 或 $\partial f(x_0)$.
- 2nd$\mathcal{O}$ (second order oracle): 适用于 Newton 法, 返回函数在 x_0 处的函数值、一和二阶导数信息 $f(x_0)$、$\nabla f(x_0)$ 和 $\nabla^2 f(x_0)$.
- $\mathcal{SFO}$ (stochastic first order oracle): 适用于随机梯度法, 返回 x_0 在 $F(x,\xi)$ 的函数值和一阶随机梯度信息 $F(x_0,\xi_0)$ 和 $G(x_i,\xi_0)$.
- $\mathcal{PO}$ (projection/prox-operator oracle): 适用于投影梯度法, 返回 x_0 在 X 上的投影

$$y\in\operatorname*{argmin}_{x\in X}\|x_0-x\|^2. \tag{1.13}$$

对于求解复合函数 $f(x)=g(x)+h(x)$ 的邻近点梯度法, 返回 x_0 的邻近点投影

$$y\in\operatorname*{argmin}_x\left\{h(x)+g(x_0)+\langle\nabla g(x_0),x-x_0\rangle+\frac{L}{2}\|x_0-x\|^2\right\}. \tag{1.14}$$

• $\mathcal{LO}$ (linear optimization oracle): 适用于条件梯度法, 当 X 是多面体时, 对给定 x_0 返回 y, 其中 y 是线性规划的解:

$$y\in\operatorname*{argmin}_{x\in X}\langle x_0,x\rangle.$$

• $\mathcal{SO}$ (separation oracle): 适用于椭球法, 对于有界闭约束集合 $X\subset\mathbb{R}^n$, 给定点 $c_0\in\mathbb{R}^n$, 如果 $c_0\in X$, 则返回真, 否则返回一个向量 w, 在 c_0 处形成一个分割超平面:

$$w^{\mathrm{T}}(x-c_0)\leqslant 0,\quad \forall x\in X. \tag{1.15}$$

上述子程序的任意组合可以形成不同算法所需要的子程序. 例如, $\mathcal{FO}+\mathcal{SO}$ 组成椭球法和重心法每一步所需要调用的子程序; $\mathcal{FO}+\mathcal{PO}$ 组成投影梯度法所需的子程序; $\mathcal{FO}+\mathcal{LO}$、$\mathcal{SFO}+\mathcal{PO}$ 和 $\mathcal{SFO}+\mathcal{LO}$ 分别组成条件梯度法、随机梯度法和随机条件梯度法所需要调用的子程序.

1.2.3 解的精度 $\mathcal{T}_\epsilon$

对于每个求解 (1.1) 的算法, 我们都会要求其求解精度. 对于不同的问题, 我们会使用不同解的精度来衡量算法的复杂度. 大致分为以下两类.

• 确定性的优化问题, 当目标函数 $f(x)$ 是凸函数时, 所求局部最优解也是全局最优解, 所以, 算法第 k 步输出 x_k 满足如下条件之一时停止算法:

$$f(x_k)-f^*\leqslant\epsilon,\quad \frac{f(x_k)-f^*}{f(x_k)}\leqslant\delta,\quad \|\nabla f(x_k)\|\leqslant\epsilon,\quad \|x_k-x^*\|\leqslant\epsilon. \tag{1.16}$$

当目标函数 $f(x)$ 是非凸函数时, 局部最优解不一定是全局最优解, 因此, 对于某一局部极小值 f^*, 会出现 $f(x_k)-f^*\leqslant 0$ 的情形, 因此不采用目标函数的误差来衡量解的精度, 而是采用 (1.16) 后两个条件, 解梯度的范数或解与最优解误差来作为停止条件.

• 随机优化问题, 由于随机优化算法的解是一个随机变量, 我们不光需要衡量解的精度, 还需要衡量解的置信度. 采取两种衡量解的条件, 其中一个是 ϵ 解:

$$\mathbb{E}[f(x_k)-f^*]\leqslant\epsilon\quad 或\quad \mathbb{E}[\|\nabla f(x_R)\|^2]\leqslant\epsilon. \tag{1.17}$$

ϵ 解衡量的是算法多次运行求得解的期望收敛效率. 另一个 (ϵ,δ) 衡量的是算法一次运行求得的解精度达到 ϵ 的置信率:

$$\mathbf{Prob}\{f(x_k)-f^*\geqslant\epsilon\}\leqslant\delta\quad 或\quad \mathbf{Prob}\{\|\nabla f(x_R)\|^2\geqslant\epsilon\}\leqslant\delta. \tag{1.18}$$

1.3 复杂度分析的算法模型

为了求解问题 $\mathcal{P}\in\mathcal{F}$, 我们可以采用迭代过程, 即迭代调用子程序的策略. 我们对梯度法和随机梯度法等迭代的优化算法的步骤进行抽象, 得到迭代算法框架, 以此来衡量优化算法在执行过程中的复杂度. 首先, 给出确定优化问题的抽象迭代算法的框架 (算法 1.1).

算法 1.1 抽象迭代算法 $\mathcal{S}$ 的框架 (确定优化问题)

输入: 给定精度 $\epsilon > 0$ 和初始点 $x_0 \in X$, 以及初始信息集合 $I_{-1} = \emptyset$.

对于 $k = 0, 1, \ldots$ 执行迭代,

1. 在 x_k 处调用子程序 $\mathcal{O}$, 获得目标函数 $f(x)$ 和约束集合 X 在 x_k 处的局部信息 $\mathcal{O}(x_k)$;
2. 更新收集的信息集合 $I_k = I_{k-1} \cup (x_k, \mathcal{O}(x_k))$;
3. 应用算法 $\mathcal{S}$ 规则处理信息集合 I_k 得到新的迭代点 x_{k+1};
4. 验证是否满足停止条件 $\mathcal{T}_\epsilon$, 如果满足, 输出 x_k, 否则 $k := k + 1$ 转步 1.

输出: $\bar{x} = \mathcal{S}(x_0)$.

对于随机优化问题, 需要对上述算法 1.1 做一些更改. 在执行上述算法框架的第 1 步时, 需首先根据 ξ 的分布随机抽样出 ξ_k (如果分布未知, 则采用 Monte Carlo 方法抽样), 然后调用子程序 $\mathcal{SFO}$ 得到局部信息 $\mathcal{SFO}(x_k, \xi_k)$. 然后执行第 2 步, 更新信息集合 $I_k = I_{k-1} \cup (x_k, \xi_k, \mathcal{SFO}(x_k, \xi_k))$. 后面的算法步与确定优化问题相同.

有了抽象迭代算法的框架, 我们可以定义解算法 $\mathcal{S}$ 和解算法集合 $\mathcal{M}$. 对于无约束的梯度法, 迭代过程为

$$x_{k+1} = x_k - \gamma_k \nabla f(x_k). \tag{1.19}$$

可以看到 x_{k+1} 是 x_k 和 $\nabla f(x_k)$ 的函数, 即

$$x_{k+1} = F_k(x_k, \nabla f(x_k)). \tag{1.20}$$

我们将 F_k 称为迭代规则函数. 实际上, 在抽象迭代算法 1.1 中,

$$x_{k+1} = F_k(x_0, \ldots, x_k, \nabla f(x_0), \ldots, \nabla f(x_k), f(x_0), \ldots, f(x_k)). \tag{1.21}$$

每一个具体的解算法 $\mathcal{S}$ 都对应着一组迭代规则函数 $F = (F_1, F_2, \ldots)$. 我们将不同 F 的集合对应的解算法 $\mathcal{S}$ 的集合记作解算法集合 $\mathcal{M}$. 例如,

$$x_{k+1} = x_0 + \mathrm{span}\{\nabla f(x_0), \nabla f(x_1), \ldots, \nabla f(x_k)\} \tag{1.22}$$

就对应一个解算法集合 $\mathcal{M}$.

1.3.1 复杂度分析的度量

有了问题模型和算法模型, 我们就可以定义算法 $\mathcal{S}$ 在问题 $\mathcal{P}$ 上的效率. 对于算法 1.1, 注意到其计算主要集中在两个迭代步骤. 一个是步 1, 调用子程序, 另一个是步 3, 更新新的迭代点. 因此, 我们可以定义两种测度来衡量算法 $\mathcal{S}$ 在问题 $\mathcal{P}$ 上的计算复杂度.

- 分析复杂度 (analytical complexity): 将问题 $\mathcal{P}$ 求解到精度 ϵ 总共所需要调用子程序 $\mathcal{O}$ 的次数.
- 算术复杂度 (arithmetical complexity): 将问题 $\mathcal{P}$ 求解到精度 ϵ 总共所需要执行的算术操作 (包括子程序 $\mathcal{O}$ 内部的操作和算法本身的操作).

比较上面两种衡量算法效率的复杂度概念, 我们发现算术复杂度更能实际体现算法的效率. 但当我们用特定算法 $\mathcal{S}$ 求解某一具体问题 $\mathcal{P} \in \mathcal{F}$ 时, 如果可以知道子程序 $\mathcal{O}$ 的复杂度, 则我们可以很容易地从分析复杂度推出算术复杂度. 因此, 本文主要研究算法 $\mathcal{S}$ 在问题集合 $\mathcal{F}$ 上的分析复杂度.

至此, 我们给了问题集合 $\mathcal{F}$、具体问题 $\mathcal{P}$ ($\mathcal{P} \in \mathcal{F}$)、解算法集 $\mathcal{M} := \mathcal{M}(\mathcal{P}, \epsilon)$ 和解算法 $\mathcal{S} := \mathcal{S}(\mathcal{P}, \epsilon)$ ($\mathcal{S} \in \mathcal{M}$) 4 个概念的定义. 记解算法 $\mathcal{S}$ 求解 $\mathcal{P}$ 的分析复杂度为 $N_{\mathcal{S}}(\mathcal{P}, \epsilon)$. 我们定义问题集合 $\mathcal{F}$ 的复杂度上界和下界如下.

• $\mathcal{F}$ 的复杂度上界: 对 $\mathcal{F}$ 某一给定的解算法 $\mathcal{S}$, $\mathcal{F}$ 的复杂度上界定义为

$$\mathrm{Compl}_{\mathcal{S}}(\epsilon) = \sup_{\mathcal{P}\in\mathcal{F}} N_{\mathcal{S}}(\mathcal{P},\epsilon). \tag{1.23}$$

• $\mathcal{F}$ 的复杂度下界: 对 $\mathcal{F}$ 某一给定的解算法集 $\mathcal{M}$, $\mathcal{F}$ 的复杂度上界定义为

$$\mathrm{Compl}(\epsilon) = \inf_{\mathcal{S}\in\mathcal{M}} \mathrm{Compl}_{\mathcal{S}}(\epsilon) := \inf_{\mathcal{S}\in\mathcal{M}} \sup_{\mathcal{P}\in\mathcal{F}} N_{\mathcal{S}}(\mathcal{P},\epsilon). \tag{1.24}$$

为了求 $\mathcal{F}$ 的复杂度上界, 我们需要找到可以求解 $\mathcal{F}$ 中所有问题 $\mathcal{P}$ 的解算法 $\mathcal{S}$. 而求解 $\mathcal{F}$ 的复杂度下界, 我们需要找到 $\mathcal{F}$ 中的一类病态问题, 使得解算法集合 $\mathcal{M}$ 中的算法的效率都很低.

分析复杂度与优化收敛性分析理论中的收敛率有一定的关系.

• 次线性收敛率 (sublinear rate): $f(x_k) - f^* \leqslant \frac{c}{\sqrt{k}}$, 其中 c 为常数, 令 $\frac{c}{\sqrt{k}} \leqslant \epsilon$, 得到 $k \geqslant \frac{c^2}{\epsilon^2}$, 对应的分析复杂度为 $\mathcal{O}(\frac{1}{\epsilon^2})$.

• 线性收敛率 (linear rate): $\|x_k - x^*\| \leqslant c(1-q)^k$, 其中 c 为常数, 令 $\leqslant c(1-q)^k \leqslant \epsilon$, 得到 $k \geqslant \frac{1}{q}(\ln c + \ln\frac{1}{\epsilon})$, 对应的分析复杂度为 $\mathcal{O}(\ln\frac{1}{\epsilon})$.

• 二阶收敛率 (quadratic rate): $\|x_{k+1} - x^*\| \leqslant c\|x_k - x^*\|^2$, 对应的分析复杂度为 $\mathcal{O}(\ln\ln\frac{1}{\epsilon})$.

1.4 复杂度分析的简单例子

复杂度分析的研究可以参见文献 [2–7] 等. 本小节利用文献 [3, 4] 中的两个例子来应用前面的复杂度分析理论.

例 1.1 (盒约束全局优化问题的复杂度上界和下界) 考虑如下全局优化问题:

$$\min_{x\in B_n} f(x), \tag{1.25}$$

其中约束集合 B_n 是一个 n 维的盒子 $B_n = \{x \in \mathbb{R}^n \mid 0 \leqslant x^{(i)} \leqslant 1,\ i = 1,\ldots,n\}$, 其中 $x^{(i)}$ 表示 x 第 i 个元素的值. 在 $\mathbb{R}^n$ 中, 我们使用 ℓ_∞ 范数来作为距离的度量. 假设函数 $f(x)$ 在 B_n 上相对于 ℓ_∞ 范数是 Lipschitz 连续的,

$$|f(x) - f(y)| \leqslant L\|x - y\|_\infty, \quad \forall\, x, y \in B_n. \tag{1.26}$$

问题模型 1.1 考虑盒约束全局优化问题模型 $\mathcal{F} = (\Sigma, \mathcal{O}, \mathcal{T}_\epsilon)$:

• 全局信息 Σ: $f(x)$ 在 B_n 上 ℓ_∞-Lipschitz 连续;

• 局部信息 $\mathcal{O}$: $\mathcal{ZO}$ 子程序, 对于任意给定的 x_0 返回函数值 $f(x_0)$;

• 解的精度 $\mathcal{T}_\epsilon$: 求近似解 $\bar{x} \in B_n$, 使得 $f(\bar{x}) - f^* \leqslant \epsilon$.

对于问题集合 $\mathcal{F}$ 中的任意一个具体优化问题 $\mathcal{P} \in \mathcal{F}$, 考虑求解它的一个简单解算法 $\mathcal{S}(p)$ (算法 1.2).

算法 1.2 无导数全局优化算法 $\mathcal{S}(p)$

输入: 盒约束集合每一维划分数 p, 问题维数 n.

1. 构造包含 $(p+1)^n$ 个点的测试点列:

$$x_{(i_1,\ldots,i_n)} = \left(\frac{i_1}{p}, \frac{i_2}{p}, \ldots, \frac{i_n}{p}\right), \tag{1.27}$$

其中 $(i_1,\ldots,i_n) \in \{0,\ldots,p\}^n$;

2. 遍历点列 $x_{(i_1,\ldots,i_n)}$, 寻找使目标函数值 $f(x_{(i_1,\ldots,i_n)})$ 最小的点, 记为 $\bar{x}$.

输出: $(\bar{x}, f(\bar{x}))$.

无导数全局优化算法 $\mathcal{S}(p)$ 将 B_n 均匀分成 $(p+1)^n$ 个网格点, 然后计算每个网格点上的函数值, 最后返回函数值最小的网格点作为 (1.25) 的近似解. 容易看到, $\mathcal{S}(p)$ 也属于我们的抽象迭代算法框架, 它需要迭代 $(p+1)^n$ 次, 全部遍历测试点列才能找到函数值最小的点. 只是它是按顺序更新新的点, 并未对收集的历史点列信息做任意实际操作. 于是, 我们可以衡量算法 $\mathcal{S}(p)$ 的效率.

定理 1.2 记 f^* 为问题模型 (1.25) 的全局最优解, 利用无导数全局优化算法 1.2 求解 (1.25), 有

$$f(\bar{x}) - f^* \leqslant \frac{L}{2p}. \tag{1.28}$$

对于问题模型 (1.25) 中的每一个具体问题 $\mathcal{P} \in \mathcal{F}$, 算法 $\mathcal{S}(p)$ 的分析复杂度满足

$$N_{\mathcal{S}(p)}(\mathcal{P}, \epsilon) \leqslant \left(\left\lfloor \frac{L}{2\epsilon} \right\rfloor + 2 \right)^n, \tag{1.29}$$

其中 $\lfloor a \rfloor$ 表示 a 的整数部分.

证明 由算法 1.2 不难看出问题模型 (1.25) 的全局最优解 x_* 要么落在测试点列上, 要么被测试点列组成的网格包含, 即存在 $(i_1, \ldots, i_n) \in \{0, \ldots, p\}^n$ 使得

$$x \equiv x_{(i_1,\ldots,i_n)} \leqslant x_* \leqslant x_{(i_1+1,\ldots,i_n+1)} \equiv y, \tag{1.30}$$

其中 $a \equiv (a^{(1)}, \ldots, a^{(n)}) \leqslant b \equiv (b^{(1)}, \ldots, b^{(n)})$, $a, b \in \mathbb{R}^n$ 当且仅当 $a^{(i)} \leqslant b^{(i)}$ 对任意 $i = 1, \ldots, n$ 成立. 注意到 $y^{(i)} - x^{(i)} = \frac{1}{p}$ 对任意 $i = 1, \ldots, n$ 成立, 且

$$x_*^{(i)} \in [x^{(i)}, y^{(i)}], \quad i = 1, \ldots, n. \tag{1.31}$$

记 $\hat{x} = (x + y)/2$, 构造点 $\tilde{x}$ 如下:

$$\tilde{x}^{(i)} = \begin{cases} y^{(i)}, & \text{如果 } x_*^{(i)} \geqslant \hat{x}^{(i)}, \\ x^{(i)}, & \text{其他.} \end{cases} \tag{1.32}$$

可以看到 $|\tilde{x}^{(i)} - x_*^{(i)}| \leqslant \frac{1}{2p}$, $i = 1, \ldots, n$. 因此,

$$\|\tilde{x} - x_*\|_\infty = \max_{1 \leqslant i \leqslant n} |\tilde{x}^{(i)} - x_*^{(i)}| \leqslant \frac{1}{2p}. \tag{1.33}$$

注意到 $\tilde{x}$ 也属于测试点列, 因此,

$$f(\bar{x}) - f(x_*) \leqslant f(\tilde{x}) - f(x_*) \leqslant L\|\tilde{x} - x_*\|_\infty \leqslant \frac{L}{2p}. \tag{1.34}$$

当 $n = 2$ 时 x、y、$\tilde{x}$、$\hat{x}$ 和 x_* 的关系如图 1. 取 $p = \lfloor \frac{L}{2\epsilon} \rfloor + 1$, 则 $p \geqslant \frac{L}{2\epsilon}$. 由定理 1.2 有

$$f(\bar{x}) - f^* \leqslant \frac{L}{2p} \leqslant \epsilon. \tag{1.35}$$

注意到我们需要调用 $(p+1)^n$ 次的函数值, 因此, 算法 $\mathcal{S}(p)$ 的分析复杂度满足 (1.29). □

可以看出算法 $\mathcal{S}(p)$ 的收敛速度与盒约束集合每一维划分点数 p 有关. 也就是与网格的划分密度有关. 复杂度分析中, 我们更关心算法的分析复杂度, 也就是 $\mathcal{S}(p)$ 在得到 ϵ 精度的近似解时, 需要调用多少次 $\mathcal{ZO}$ 子程序. 于是有如下关于 $\mathcal{F}$ 复杂度上界的推论.

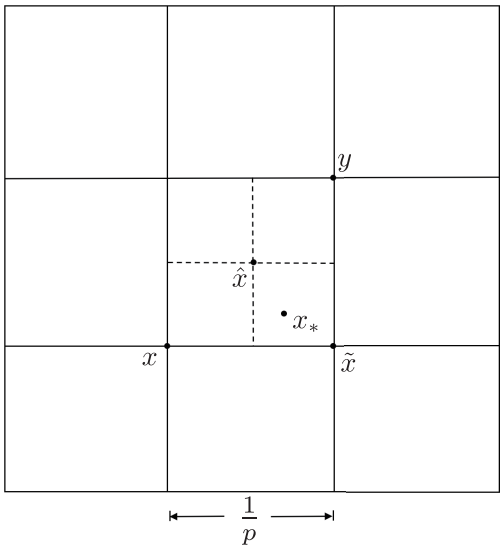

图 1 当 $n=2$ 时, x、y、$\tilde{x}$、$\hat{x}$ 和 x_* 示意图

在 (1.29) 两边关于问题集合 $\mathcal{F}$ 取极大, 我们可以得到问题集合 $\mathcal{F}$ 的复杂度上界的一个估值:

$$\mathrm{Compl}_{\mathcal{S}(p)}(\epsilon)=\max_{\mathcal{P}\in\mathcal{F}}N_{\mathcal{S}(p)}(\mathcal{P},\epsilon)\leqslant\left(\left\lfloor\frac{L}{2\epsilon}\right\rfloor+2\right)^n. \tag{1.36}$$

也就是说, 存在一个算法 (如 $\mathcal{S}(p)$) 在解决 $\mathcal{F}$ 中每一个具体问题 $\mathcal{P}$ 时 (近似解满足 $f(\bar{x})-f^*\leqslant\epsilon$, $\bar{x}\in B_n$), 所需要调用 $\mathcal{ZO}$ 子程序的次数至多不超过 $(\lfloor\frac{L}{2\epsilon}\rfloor+2)^n$.

关于结论 (1.36), 我们会提出一些问题. 第一, 我们在估计算法 $\mathcal{S}(p)$ 时太过粗略, 是否存在比 (1.36) 更好的界? 第二, 是否存在其他算法, 其效率比 $\mathcal{S}(p)$ 要好? 要回答这两个问题, 我们就需要推导问题集合 $\mathcal{F}$ 的复杂度下界.

我们首先需要定义问题集合 $\mathcal{F}$ 的解算法集合 $\mathcal{M}$.

定理 1.3 令 $\epsilon<\frac{1}{2}L$, 对于 (1.25) 对应的问题集合 $\mathcal{F}$ 来说, 对其中每一个具体问题 $\mathcal{P}\in\mathcal{F}$, 其解算法集合 $\mathcal{M}:=\mathcal{M}(\mathcal{F},\mathcal{ZO})$ 中的任意一个算法 $\mathcal{S}\in\mathcal{M}$ 的分析复杂度满足

$$\left(\left\lfloor\frac{L}{2\epsilon}\right\rfloor\right)^n\leqslant N_{\mathcal{S}}(\mathcal{P},\epsilon). \tag{1.37}$$

证明 考虑 $\mathcal{ZO}$ 子程序定义的解算法集合 $\mathcal{M}:=\mathcal{M}(\mathcal{F},\mathcal{ZO})$ 中的任意一个解算法 $\mathcal{S}$. $\mathcal{S}$ 通过对约束集合 B_n 进行采样得到测试点列 $\{x_k\}$, 然后在每个测试点列上调用 $\mathcal{ZO}$ 子程序, 通过处理 (排序) 子程序返回的信息 (函数值信息) 得到最优近似解. $\mathcal{M}$ 中解算法之间的差异仅在于测试点列的选取方式, 例如, 算法 1.2 是在 B_n 上采取均匀采样的方式. 我们利用反证法证明定理 1.3.

若存在 $\mathcal{M}$ 中的一个解算法 $\mathcal{S}_0$ 在求解问题模型 1.1 得到 ϵ 精度时调用 $\mathcal{ZO}$ 子程序的次数 $N<p^n$, 令采样的测试点列个数 $p=\lfloor\frac{L}{2\epsilon}\rfloor\geqslant 1$. 我们只需要证明存在某一目标函数, 使得 $\mathcal{S}_0$ 在求解该目标函数时的精度大于等于 ϵ.

由于 $\mathcal{S}_0$ 在 B_n 上采样的测试点列个数 $N<p^n$, 因此存在某一非测试点 $\hat{x}$ 和集合

$$B=\left\{x\;\middle|\;\hat{x}\leqslant x\leqslant\hat{x}+\frac{1}{p}e\right\}\subseteq B_n,$$

使得 B 中不包含任意测试点. 令 $x_* = \hat{x} + \frac{1}{2p}e$, 在集合 B 上构造函数

$$f_p(x) = \min\{0, L\|x - x_*\|_\infty - \epsilon\}, \quad \text{当 } x \in B. \tag{1.38}$$

当 $x \in B_n \setminus B$, 令 $f_p(x) = 0$ 时, 注意到 $\|x_* - \hat{x}\|_\infty \geqslant \frac{\epsilon}{L}$, 因此, $f_p(\hat{x}) = 0$ 的图像如图 2 所示. 注意到 $f_p(x)$ 是 ℓ_∞-Lipschitz 连续 (Lipschitz 常数为 L), 全局最优解为 $f_p(x_*) = -\epsilon$. 利用 $\mathcal{S}_0$ 求解 $f_p(x)$ 时, 由于在所要测试点列处调用 $\mathcal{ZO}$ 子程序都返回 $f_p(x_k) = 0$, 因此求得近似最优解为 $f_p(\bar{x}) = 0$, 于是有

$$f_p(\bar{x}) - f_p(x_*) \geqslant \epsilon. \tag{1.39}$$

因此得到结论: 若测试点列的个数 $N < (\lfloor \frac{L}{2\epsilon} \rfloor)^n$ (子程序 $\mathcal{ZO}$ 的调用次数), 则对应的解算法求得的精度不可能比 ϵ 更好, 即问题模型 1.1 的复杂度下界为 $(\lfloor \frac{L}{2\epsilon} \rfloor)^n$. □

例 1.2 (一维凸优化问题的复杂度上界和下界)　考虑一维凸优化问题

$$\min_{x \in [a,b]} \ f(x). \tag{1.40}$$

问题模型 1.2　考虑一维凸优化问题模型 $\mathcal{F} = (\Sigma, \mathcal{O}, \mathcal{T}_\epsilon)$:

• 全局信息 Σ: 求优化问题 $\min\{f(x) \mid x \in [a,b]\}$, $f(x)$ 是凸函数、连续, 且满足 $0 \leqslant f(x) \leqslant V$, $a < b$ 是任意常数;

• 局部信息 $\mathcal{O}$: $\mathcal{FO}$ 子程序, 对于任意给定的 x_0 返回函数值 $f(x_0)$ 和 x_0 处次梯度 $\partial f(x_0)$;

• 解的精度 $\mathcal{T}_\epsilon$: 求近似解 $\bar{x} \in [a,b]$, 使得 $f(\bar{x}) - f^* \leqslant \epsilon$.

文献 [4] 给出了问题模型 1.2 的复杂度上下界.

定理 1.4　对于问题模型 1.2 来说, 其复杂度满足

$$\left\lceil \frac{1}{5} \log_2 \left(\frac{V}{\epsilon} \right) \right\rceil \leqslant N(\epsilon) \leqslant \left\lfloor \log_2 \left(\frac{V}{\epsilon} \right) \right\rfloor. \tag{1.41}$$

证明　复杂度上界: 为了求得问题模型 1.2 的复杂度上界, 我们只需要提出问题模型 1.2 的一个解算法 $\mathcal{S}(\epsilon)$, 它求解问题模型 1.2 时的复杂度就是该问题模型的复杂度上界. 我们利用最简单的二分

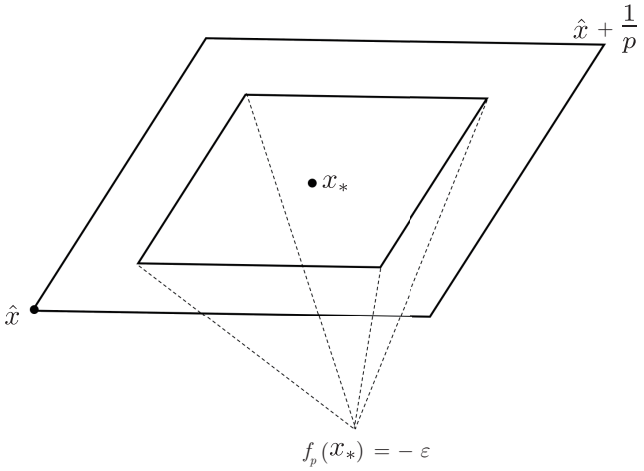

图 2　$f_p(x)$ 在集合 B 上的示意图

法来作为解算法, 其求解步骤如下: 记目标集合为 $X_k=[l_k,u_k]$, 迭代点为 x_k, 取 $X_0=[a,b]$, 即 $l_0=a$, $u_0=b$, 对 $k=1,\ldots,N$ 执行迭代,

(1) 更新 $x_k=(l_{k-1}+u_{k-1})/2$;

(2) 更新 X_k: 在 x_k 处调用子程序得到次梯度 $f'(x_k)$, 按图 3 所示三种情形: 若 $f'(x_k)<0$, 则 $l_k=l_{k-1}$, $u_k=x_k$; 若 $f'(x_k)=0$, 则输出 $\bar{x}=x_k$, 停止迭代; 若 $f'(x_k)>0$, 则 $l_k=x_k$, $u_k=u_{k-1}$. 最后输出解 $\bar{x}\in\{x_1,\ldots,x_N\}$ 满足

$$f(\bar{x})=\min_{1\leqslant i\leqslant N}f(x_i). \tag{1.42}$$

首先可以看到迭代集合 X_k 的长度为 $(b-a)/2^k$. 同时对任意的 $x\in X\setminus X_k$,

$$f(x)\geqslant f(\bar{x})=\min_{1\leqslant i\leqslant N}f(x_i)$$

成立. 取 α 满足 $2^{-N}<\alpha<1$, 构造约束集合 $X=[a,b]$ 的 α 倍收缩集合,

$$X_\alpha=(1-\alpha)x^*+\alpha X\equiv\{(1-\alpha)x^*+\alpha z\mid z\in X\}. \tag{1.43}$$

不难证明 X_α 是 x^* 处长度为 $\alpha(b-a)$ 的线段. 考虑到 α 的取值, 可以知道 X_α 无法包含在 X_N 中, 即存在 $y\in X_\alpha$ 使得 $y\notin X_N$. 设 $y=(1-\alpha)x^*+\alpha z$ 对某一 $z\in X$ 成立. 由 f 的凸性, 有

$$f(y)\leqslant(1-\alpha)f(x^*)+\alpha f(z). \tag{1.44}$$

可以推出

$$f(y)-f(x^*)\leqslant\alpha(f(z)-f^*)\leqslant\alpha V. \tag{1.45}$$

因此, y 是原问题的一个 αV 精度的近似解. 注意到 $y\notin X_N$, 因此有 $f(\bar{x})\leqslant f(y)$, 即

$$f(\bar{x})-f^*\leqslant f(y)-f^*\leqslant\alpha V. \tag{1.46}$$

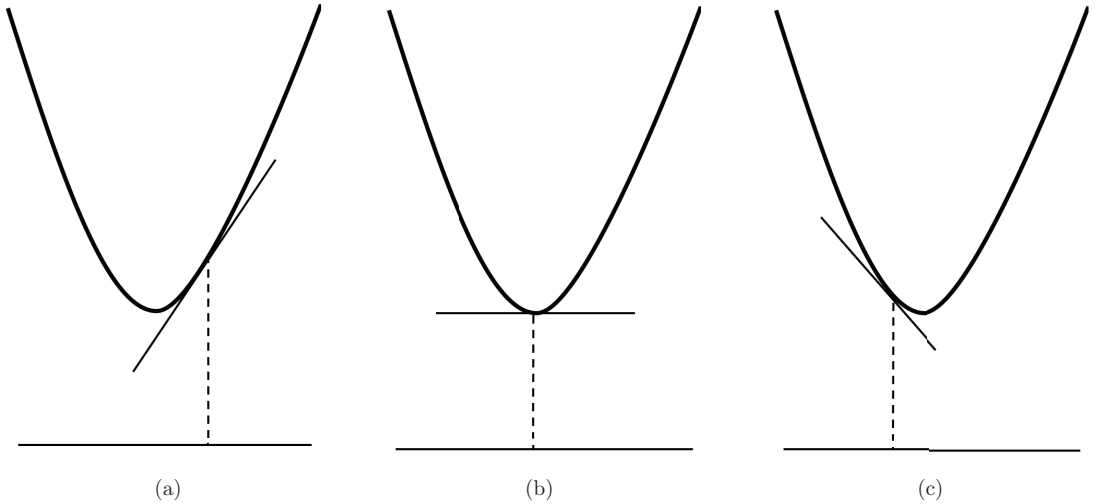

图 3 二分法示意图

由于 α 可以无限接近 2^{-N}, 因此取 $N=\lfloor\log_2(\frac{V}{\epsilon})\rfloor$, 有

$$f(\bar{x})-f^*\leqslant 2^{-N}V\leqslant 2^{-\lfloor\log_2(\frac{V}{\epsilon})\rfloor}V\leqslant\epsilon. \tag{1.47}$$

我们证明了问题模型 1.2 的复杂度上界为 $\lfloor\log_2(\frac{V}{\epsilon})\rfloor$.

复杂度下界: 与证明复杂度上界不同, 复杂度下界的证明需要构造问题模型 1.2 中一个最坏情形的函数. 不失一般性, 令 $X=[-1,1]$, $V=1$, 则我们需要证明对任意给定的精度 $\epsilon\in(0,1)$, 问题集合 1.2 的复杂度都不小于

$$\left\lceil\frac{1}{5}\log_2\left(\frac{1}{\epsilon}\right)\right\rceil. \tag{1.48}$$

也就是说, 我们需要证明对于问题模型 1.2 任意的解算法 $\mathcal{S}(\epsilon)$, 总能构造问题模型 1.2 中的一个函数, 使得 $\mathcal{S}(\epsilon)$ 在求解该函数时的分析复杂度都不小于 (1.48).

令 $N=\lceil\frac{1}{5}\log_2(\frac{1}{\epsilon})\rceil$, 对任意的 $0\leqslant i\leqslant N$ 构造函数 $f_i(x)$ 满足如下性质.

(1) $f_i(x)$ 凸、连续且存在长度为 $\delta_i=2^{1-2i}$ 的子集合 $\Delta_i\subseteq X=[-1,1]$ 使得

$$f_i(x)=a_i+2^{-3i}|x-c_i|,\quad x\in\Delta_i, \tag{1.49}$$

其中 c_i 是 Δ_i 的中点.

(2) 解算法 $\mathcal{S}(\epsilon)$ 求解 $f_i(x)$ 时产生的前 i 个点列 $x_1,\ldots,x_i$ 落在 $X\setminus\Delta_i$ 上. 我们用递推法证明存在这样的函数序列 $f_i(x)$. 当 $i=0$ 时, 取

$$f_0(x)=|x|,\quad \Delta_0=X=[-1,1], \tag{1.50}$$

满足性质 (1) 和 (2). 假设存在 f_i 和 Δ_i 满足性质 (1) 和 (2), 我们利用 f_i 和 Δ_i 来构造 f_{i+1} 和 Δ_{i+1} . 令 $x_1,\ldots,x_{i+1}$ 表示解算法 $\mathcal{S}(\epsilon)$ 求解 $f_i(x)$ 时产生的前 $i+1$ 个点列. 由假设可知搜索点列 $x_1,\ldots,x_i$ 落在 $X\setminus\Delta_i$ 上, 我们根据 x_{i+1} 的位置构造 f_{i+1}, 满足如下性质:

(1) 当 $x\in X\setminus\Delta_i$ 时 $f_{i+1}(x)=f_i(x)$;

(2) 若 x_{i+1} 落在 c_i 右边, 取 c_{i+1} 为 Δ_i 左半边中点, 构造如图 4 的函数 f_{i+1} 和 Δ_{i+1}. 其中 $\overline{ab}$、$\overline{bc_{i+1}}$、$\overline{c_{i+1}d}$ 和 $\overline{c_ie}$ 的斜率依次为 -2^{-3i}、$-2^{-3(i+1)}$、$2^{-3(i+1)}$ 和 2^{-3i};

(3) 若 x_{i+1} 落在 c_i 左边, 构造方式与落在右边对称.

我们可以得到满足性质的 f_{i+1} 和 Δ_{i+1}. 注意到 f_i 是凸函数, 可以知道 c_i 是 f_i 的全局最优解. 考虑 f_N 和 Δ_N, 由于 $x_1,\ldots,x_N$ 都落在 $X\setminus\Delta_N$ 上, 因此, 解算法 $\mathcal{S}(\epsilon)$ 求解 $f_N(x)$ 时产生的解 $\bar{x}(\mathcal{S},f_N)$ (由 $x_1,\ldots,x_N$ 得到) 与最优解 c_N 之间的函数值差至少为 $2^{-3N}\times 2^{-2N}=2^{-5N}$ (线段 $\overline{ac_N}$ 的垂直高度). 因此,

$$f_N(\bar{x}(\mathcal{S},f_N))-f_K^*>2^{-5N}. \tag{1.51}$$

令 $2^{-5N}<\epsilon$, 得到 $N>\frac{1}{5}\log(\frac{1}{\epsilon})$. 我们得到问题模型 1.2 的算法复杂度下界至少为 (1.48). □

1.5 算法复杂度表

我们将文献中研究的优化算法求解确定性优化问题的收敛性和复杂度列在表 1 中, 其中 gravity 代表重心法, ellipsoid 代表椭球法, PGD (projected gradient) 代表投影梯度法, AGD (accelerated gradient) 代表加速梯度法, CndG (conditional gradient) 代表条件梯度法, CGS (conditional gradient sliding) 代表加速条件梯度法. 表中求解的函数都是凸函数, $X\subseteq\mathbb{R}^n$ 是闭且凸的满足 $\mathcal{B}(r)\subseteq X\subseteq\mathcal{B}(R)$. 其中

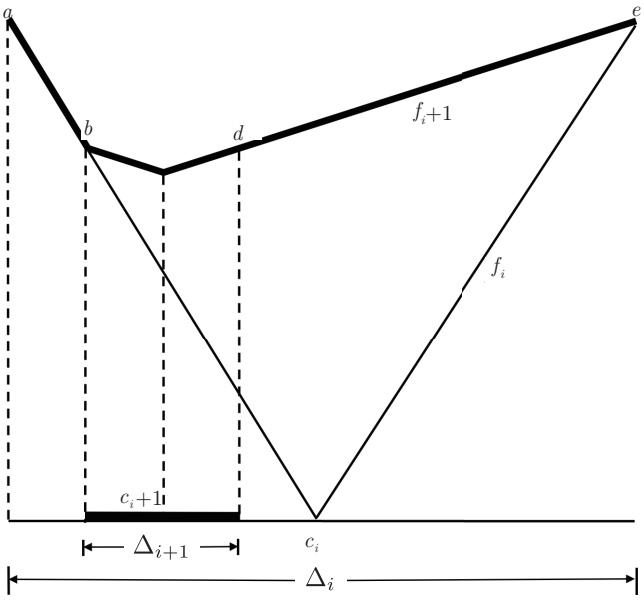

图 4 由 f_i 构造 f_{i+1} 示意图

$Q = L/\mu$, L 表示梯度的 Lipschitz 常数, μ 表示强凸函数对应的常数, 表中重心法、椭球法和 AGD 算法求解 $C_L^{1,\alpha}(\mathbb{R}^n)$ 的复杂度结论来自文献 [2, 4–6]. PGD 算法求解 $\mathcal{S}_{L,\mu}^{1,1}(X)$ 和 $\mathcal{W}_{L,\mu}^{1,1}(X)$ 函数类的复杂度结论来自文献 [1]. CSG 和 CndG 算法的复杂度结论来自文献 [8–10]. 其余 PGD 和 AGD 算法求解凸函数类的复杂度结论来自于文献 [3, 4, 11–17].

表 1 确定性优化问题的复杂度分析表

问题集合 $\mathcal{F}$	算法 $\mathcal{S}$	子程序 $\mathcal{O}$	收敛速率	分析复杂度
$C^0(X)$	gravity	$\mathcal{FO}+\mathcal{SO}$	$\exp(-\frac{k}{n})$	$n\log(\frac{B}{\epsilon})$
$C^0(X)$	ellipsoid	$\mathcal{FO}+\mathcal{SO}$	$\frac{R}{r}\exp(-\frac{k}{n^2})$	$n^2\log(\frac{BR}{r\epsilon})$
$C_L^{0,1}(X)$	PGD	$\mathcal{FO}+\mathcal{PO}$	$\frac{LR}{\sqrt{k}}$	$\frac{L^2R^2}{\epsilon^2}$
$\mathcal{F}_{L,\mu}^{0,1}(X)$	PGD	$\mathcal{FO}+\mathcal{PO}$	$\frac{L^2}{\mu k}$	$\frac{L^2}{\mu\epsilon}$
$C_L^{1,1}(X)$	PGD	$\mathcal{FO}+\mathcal{PO}$	$\frac{LR^2}{k}$	$\frac{LR^2}{\epsilon}$
$C_L^{1,1}(X)$	AGD	$\mathcal{FO}+\mathcal{PO}$	$\frac{LR^2}{k^2}$	$\frac{\sqrt{L}R}{\sqrt{\epsilon}}$
$C_L^{1,1}(X)$	CndG	$\mathcal{FO}+\mathcal{LO}$	$\frac{LR^2}{k}$	$\frac{LR^2}{\epsilon}$
$C_L^{1,1}(X)$	CGS	$\mathcal{FO}+\mathcal{LO}$	$\frac{LR^2}{k^2}$	$\mathcal{FO}:\sqrt{LR^2/\epsilon},\ \mathcal{LO}:\frac{LR^2}{\epsilon}$
$\mathcal{S}_{L,\mu}^{1,1}(X)$	PGD	$\mathcal{FO}+\mathcal{PO}$	$LR^2(\frac{Q}{Q+1})^k$	$\log(\frac{LR^2}{\epsilon})/\log(\frac{Q+1}{Q})$
$\mathcal{W}_{L,\mu}^{1,1}(X)$	PGD	$\mathcal{FO}+\mathcal{PO}$	$LR^2(\frac{Q-1}{Q+1})^k$	$\log(\frac{LR^2}{\epsilon})/\log(\frac{Q+1}{Q-1})$
$\mathcal{F}_{L,\mu}^{1,1}(X)$	PGD	$\mathcal{FO}+\mathcal{PO}$	$LR^2(\frac{Q-1}{Q+1})^{2k}$	$\log(\frac{LR^2}{\epsilon})/\log(\frac{Q+1}{Q-1})^2$
$\mathcal{F}_{L,\mu}^{1,1}(X)$	AGD	$\mathcal{FO}+\mathcal{PO}$	$LR^2(\frac{\sqrt{Q}-1}{\sqrt{Q}})^k$	$\log(\frac{LR^2}{\epsilon})/\log(\frac{\sqrt{Q}}{\sqrt{Q}-1})$
$\mathcal{F}_{L,\mu}^{1,1}(X)$	CndG	$\mathcal{FO}+\mathcal{LO}$	$\mu R/2^t$	$Q\log(\frac{\mu R}{\epsilon})$
$\mathcal{F}_{L,\mu}^{1,1}(X)$	CGS	$\mathcal{FO}+\mathcal{LO}$	$\delta_0/2^t$	$\mathcal{FO}:\sqrt{Q}\log\frac{\delta_0}{\epsilon},\ \mathcal{LO}:\frac{LR^2}{\epsilon}$
$C_L^{1,\alpha}(\mathbb{R}^n)$	AGD	$\mathcal{FO}$	$\frac{2LR^{1+\alpha}}{(1+3\alpha)\log k}$	$(\frac{LR^{1+\alpha}}{\epsilon})^{(2/1+3\alpha)}$

2 光滑优化问题的复杂度分析

2.1 引言

本节给出光滑优化问题的复杂度分析. 考虑优化问题

$$f^* = \min_{x\in X} f(x). \tag{2.1}$$

本节考虑优化问题 (2.1) 中目标函数 $f(x)$ 是光滑的情形. 首先定义光滑函数的一些子集, 并列出它们的一系列性质, 用于收敛性分析. 这些性质的证明可以在文献 [11] 中找到.

定义 2.1 (光滑函数子集合) 设 X 为 $\mathbb{R}^n$ 的一个子集, 记 $C_L^{k,p}(X)$ 为具有如下性质的函数集合:
(1) 任意函数 $f \in C_L^{k,p}(X)$ 在 X 上 k 次连续可微;
(2) 任意函数 $f \in C_L^{k,p}(X)$ 的 p 阶导数在 X 上 Lipschitz 连续 (对于某常数 L),

$$\|f^{(p)}(x) - f^{(p)}(y)\| \leqslant L\|x-y\|, \quad \text{对任意 } x, y \in X. \tag{2.2}$$

记 $\mathcal{F}_L^{k,p}(X)$ 表示函数集合 $C_L^{k,p}(X)$ 与凸函数集合的交集.

性质 2.1 若 $f_1 \in C_{L_1}^{k,p}(X)$, $f_2 \in C_{L_2}^{k,p}$, 且 $\alpha, \beta \in \mathbb{R}^1$, 则对 $L_3 = |\alpha|L_1 + |\beta|L_2$, 有 $\alpha f_1 + \beta f_2 \in C_{L_3}^{k,p}(X)$.

性质 2.2 $f \in C_L^{2,1}(\mathbb{R}^n) \subset C_L^{1,1}(\mathbb{R}^n)$ 当且仅当对任意 $x \in \mathbb{R}^n$, $\|\nabla^2 f(x)\| \leqslant L$.

性质 2.3 令 $f \in C_L^{1,1}(\mathbb{R}^n)$, 则对任意 $x, y \in \mathbb{R}^n$, 有

$$|f(y) - f(x) - \langle \nabla f(x), y-x\rangle| \leqslant \frac{L}{2}\|y-x\|^2. \tag{2.3}$$

性质 2.4 令 $f \in C_L^{2,2}(\mathbb{R}^n)$, 则对任意 $x, y \in \mathbb{R}^n$, 有

$$\|\nabla f(y) - \nabla f(x) - \langle \nabla^2 f(x), y-x\rangle\| \leqslant \frac{L}{2}\|y-x\|^2, \tag{2.4}$$

$$\left|f(y) - f(x) - \langle \nabla f(x), y-x\rangle - \frac{1}{2}\langle \nabla^2 f(x)(y-x), y-x\rangle\right| \leqslant \frac{L}{6}\|y-x\|^3. \tag{2.5}$$

性质 2.5 令 $f \in C_L^{2,2}(\mathbb{R}^n)$ 且 $\|y-x\| = r$, 则有

$$\nabla^2 f(x) - LrI_n \preceq \nabla^2 f(y) \preceq \nabla^2 f(x) + LrI_n. \tag{2.6}$$

性质 2.6 令 $f \in \mathcal{F}_L^{1,1}(\mathbb{R}^n)$, 则对任意 $x, y \in \mathbb{R}^n$, $\alpha \in [0,1]$, 下列不等式成立:

$$0 \leqslant |f(y) - f(x) - \langle \nabla f(x), y-x\rangle| \leqslant \frac{L}{2}\|y-x\|^2, \tag{2.7}$$

$$f(x) + \langle \nabla f(x), y-x\rangle + \frac{1}{2L}\|\nabla f(x) - \nabla f(y)\|^2 \leqslant f(y), \tag{2.8}$$

$$\frac{1}{L}\|\nabla f(y) - \nabla f(x)\|^2 \leqslant \langle \nabla f(x) - \nabla f(y), x-y\rangle, \tag{2.9}$$

$$\langle \nabla f(x) - \nabla f(y), x-y\rangle \leqslant L\|x-y\|^2, \tag{2.10}$$

$$\alpha f(x) + (1-\alpha)f(y) \leqslant f(\alpha x + (1-\alpha)y) + \frac{\alpha(1-\alpha)}{2L}\|\nabla f(x) - \nabla f(y)\|^2, \tag{2.11}$$

$$\alpha f(x) + (1-\alpha)f(y) \leqslant f(\alpha x + (1-\alpha)y) + \alpha(1-\alpha)\frac{L}{2}\|x-y\|^2. \tag{2.12}$$

2.2 复杂度上界

本小节分析不同算法在不同光滑函数子集合中的收敛性和复杂度, 并给出不同光滑函数子集合的复杂度上界. 考虑 $X=\mathbb{R}^n$ 情形的优化问题

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

我们分析梯度算法 (算法 2.1) 在求解上述问题时的复杂度.

算法 2.1 梯度法

输入: $x_0\in\mathbb{R}^n$, 步长序列 $\{h_k\}$.
对 $k=0,1,\ldots$ 执行迭代,

$$x_{k+1}=x_k-h_k\nabla f(x_k). \tag{2.13}$$

梯度法的步长 h_k 有如下 3 种选择方式:
(1) 固定步长和变步长策略: 取固定步长 $h_k=h>0$, 取变步长 $h_k=\frac{h}{\sqrt{k+1}}$;
(2) 精确先搜索步长法: $h_k=\operatorname{argmin}_{h\geqslant 0} f(x_k-h\nabla f(x_k))$;
(3) Goldenstein-Armijo 准则: 令 $x_{k+1}=x_k-h_k\nabla f(x_k)$, 寻找 h_k 满足不等式

$$\alpha\langle\nabla f(x_k),x_k-x_{k+1}\rangle\leqslant f(x_k)-f(x_{k+1}), \tag{2.14}$$

$$\beta\langle\nabla f(x_k),x_k-x_{k+1}\rangle\geqslant f(x_k)-f(x_{k+1}), \tag{2.15}$$

其中 $0<\alpha<\beta<1$ 为固定常数.

下面列出算法 2.1 在求解无约束和约束优化问题时的复杂度结论.

问题模型 2.1 考虑无约束非凸优化问题集合 $\mathcal{F}=(\Sigma,\mathcal{O},\mathcal{T}_\epsilon)$:

• 全局信息 Σ: 目标函数 $f\in C_L^{1,1}(\mathbb{R}^n)$, 且 $f(x)$ 不一定是凸函数; $f(x)$ 下有界 (存在常数 M 使得 $f(x)\geqslant M$, $\forall\, x\in\mathbb{R}^n$), 约束集合 $X\equiv\mathbb{R}^n$;

• 局部信息 $\mathcal{O}$: $\mathcal{FO}$ 子程序, 对于任意给定的 x_0 返回函数值 $f(x_0)$ 和一阶导数 $\nabla f(x_0)$;

• 解的精度 $\mathcal{T}_\epsilon$: 求局部极小值的近似解 $\bar{x}\in\mathbb{R}^n$, 使得 $\|\nabla f(\bar{x})\|\leqslant\epsilon$.

定理 2.1 算法 2.1 在求解问题模型 2.1 时, 取 $\bar{x}$ 满足 $\|\nabla f(\bar{x})\|=\min_{0\leqslant k\leqslant N}\|\nabla f(x_k)\|$, 则算法的收敛速度为

$$\|\nabla f(\bar{x})\|\leqslant\frac{1}{\sqrt{N+1}}\left[\frac{L}{\omega}(f(x_0)-f^*)\right]^{1/2}. \tag{2.16}$$

问题模型 2.1 的分析复杂度上界为

$$N(\epsilon)\leqslant\frac{L(f(x_0)-f^*)}{\omega\epsilon^2}, \tag{2.17}$$

其中 ω 为常数.

证明 首先, 我们可以估计梯度法迭代一步的下降量. 对于固定步长 $h_k\equiv h$, 在性质 2.3 中取 $y=x_{k+1}=x_k-h\nabla f(x_k)$ 和 $x=x_k$, 有

$$\begin{aligned}f(x_{k+1})&\leqslant f(x_k)+\langle\nabla f(x_k),x_{k+1}-x_k\rangle+\frac{L}{2}\|x_{k+1}-x_k\|^2\\&=f(x_k)-h\|\nabla f(x_k)\|^2+\frac{h^2}{2}L\|\nabla f(x_k)\|^2\end{aligned}$$

$$= f(x_k) - h\left(1 - \frac{h}{2}L\right)\|\nabla f(x_k)\|^2. \tag{2.18}$$

将 (2.18) 右边关于 h 求极小, 我们可以得到梯度法迭代一步时, 目标函数的最优下降量

$$\min_{h>0}\left\{f(x_k) - h\left(1 - \frac{h}{2}L\right)\|\nabla f(x_k)\|^2\right\} = f(x_k) - \frac{1}{2L}\|\nabla f(x_k)\|^2 \tag{2.19}$$

当 $h^* = \frac{1}{L}$ 时取到最小值. 因此, 取恒定步长 $h \equiv \frac{1}{L}$, 则梯度法迭代一步的估计为

$$f(x_k) - f(x_{k+1}) \geqslant \frac{1}{2L}\|\nabla f(x_k)\|^2. \tag{2.20}$$

同样地, 我们可以取固定步长 $h_k \equiv \frac{2\alpha}{L}$, $\alpha \in (0,1)$, 则

$$f(x_k) - f(x_{k+1}) \geqslant \frac{2}{L}\alpha(1-\alpha)\|\nabla f(x_k)\|^2. \tag{2.21}$$

对于精确线搜索步长, 下降量不会比固定步长 $h_k \equiv \frac{1}{L}$ 差, 因此也有下降量不等式 (2.20). 对于 Goldenstein-Armijo 准则, 结合不等式 (2.15) 和 (2.18), 有

$$\beta\langle\nabla f(x_k), x_k - x_{k+1}\rangle \geqslant f(x_k) - f(x_{k+1}) \geqslant h_k\left(1 - \frac{h_k}{2}L\right)\|\nabla f(x_k)\|^2. \tag{2.22}$$

由于 $\beta\langle\nabla f(x_k), x_k - x_{k+1}\rangle = \beta h_k\|\nabla f(x_k)\|^2 \geqslant h_k(1 - \frac{h_k}{2}L)\|\nabla f(x_k)\|^2$, 我们得到 $h_k \geqslant \frac{2}{L}(1-\beta)$, 结合不等式 (2.14) 得到

$$f(x_k) - f(x_{k+1}) \geqslant \alpha h_k\|(x_k)\|^2 \geqslant \frac{2\alpha(1-\beta)}{L}\|\nabla f(x_k)\|^2. \tag{2.23}$$

综上, 梯度法的 3 种步长选择都有如下下降量估计:

$$f(x_k) - f(x_{k+1}) \geqslant \frac{\omega}{L}\|\nabla f(x_k)\|^2. \tag{2.24}$$

现在可以估计梯度法的复杂度了, 将上述不等式对 $k = 0, \ldots, N$ 求和, 得到

$$\frac{\omega}{L}\sum_{k=0}^{N}\|\nabla f(x_k)\|^2 \leqslant f(x_0) - f(x_{N+1}) \leqslant f(x_0) - f^*. \tag{2.25}$$

由于上式右边是常数, 因此, 当 $N \to \infty$ 时, $\|\nabla f(x_N)\| \to 0$. 注意到 $\|\nabla f(\bar{x})\| = \min_{0\leqslant k\leqslant N}\|\nabla f(x_k)\|$, 利用不等式 (2.25) 可以得到不等式 (2.16). □

注意到梯度法求解该问题模型时, 表现为全局次线性收敛, 分析复杂度上界为 $\mathcal{O}(\frac{L}{\epsilon^2})$.

问题模型 2.2 考虑无约束非凸优化问题集合 $\mathcal{F} = (\Sigma, \mathcal{O}, \mathcal{T}_\epsilon)$:

• 全局信息 Σ: 目标函数 $f \in C_L^{2,2}(\mathbb{R}^n)$, 且 $f(x)$ 不一定是凸函数; $f(x)$ 存在局部极小点 x^*, 且 $\nabla^2 f(x^*)$ 正定; 存在常数 $0 < m \leqslant M < \infty$ 使得 $mI_n \preceq \nabla^2 f(x^*) \preceq MI_n$; 初始点 x_0 距离 x^* 足够近; 约束集合 $X \equiv \mathbb{R}^n$;

• 局部信息 $\mathcal{O}$: $\mathcal{FO}$ 子程序, 对于任意给定的 x_0 返回函数值 $f(x_0)$ 和一阶导数 $\nabla f(x_0)$;

• 解的精度 $\mathcal{T}_\epsilon$: 求局部极小值的近似解 $\bar{x} \in \mathbb{R}^n$, 使得 $\|\bar{x} - x^*\| \leqslant \epsilon$.

定理 2.2 假设梯度法的初始点 x_0 距离局部极小点 x^* 足够近, 满足 $r_0 = \|x_0 - x^*\| < \bar{r} = \frac{2m}{L}$, 且步长取 $h_k \equiv \frac{2}{m+M}$, 则梯度法求解问题模型 2.2 的收敛速率为

$$\|x_k - x^*\| \leqslant \frac{\bar{r}r_0}{\bar{r} - r_0}\left(1 - \frac{2m}{M+3m}\right)^k. \tag{2.26}$$

复杂度上界为

$$\frac{M+3m}{2m}\left[\ln\left(\frac{\bar{r}r_0}{\bar{r}-r_0}\right)+\ln\frac{1}{\epsilon}\right]. \tag{2.27}$$

该定理的证明可以在文献 [11] 中找到. 注意到梯度法求解该问题模型时, 表现为局部线性收敛, 且复杂度上界为 $\mathcal{O}(\ln\frac{1}{\epsilon})$.

考虑定义 1.1 中强凸函数集合 $\mathcal{F}_{L,\mu}^{1,1}(\mathbb{R}^n)$, 若 $f\in\mathcal{F}_{L,\mu}^{1,1}(\mathbb{R}^n)$, 则 $f\in C_L^{1,1}(\mathbb{R}^n)$ 且 f 是强凸函数, 即满足

$$f(y)\geqslant f(x)+\langle\nabla f(x),y-x\rangle+\frac{\mu}{2}\|x-y\|^2. \tag{2.28}$$

我们可以将强凸函数的条件放松, 得到弱强凸函数和二阶增长函数. 记 X^* 表示凸函数 $f(x)$ 在 $\mathbb{R}^n$ 上最小值对应的点组成的集合. 文献 [1] 给出了弱强凸函数集合 $\mathcal{W}_{L,\mu}^{1,1}(\mathbb{R}^n)$ 和二阶增长函数集合 $\mathcal{S}_{L,\mu}^{1,1}(\mathbb{R}^n)$ 的定义, 并分析梯度法在该函数集合上的复杂度上界.

考虑弱强凸函数 $\mathcal{W}_{L,\mu}^{1,1}(\mathbb{R}^n)$, 若 $f\in\mathcal{W}_{L,\mu}^{1,1}(\mathbb{R}^n)$, 则 $f\in C_L^{1,1}(\mathbb{R}^n)$ 且对任意 $x\in\mathbb{R}^n$ 满足

$$f^*\geqslant f(x)+\langle\nabla f(x),\bar{x}-x\rangle+\frac{\mu}{2}\|x-\bar{x}\|^2, \tag{2.29}$$

其中 $\bar{x}=\operatorname{argmin}_{y\in X^*}\|y-x\|$, 即 x 在 X^* 的投影. 可以看到弱强凸函数只是在最优点处满足不等式 (2.28), 因此, $\mathcal{F}_{L,\mu}^{1,1}(\mathbb{R}^n)\subset\mathcal{W}_{L,\mu}^{1,1}(\mathbb{R}^n)$.

考虑二阶增长函数集合 $\mathcal{S}_{L,\mu}^{1,1}(\mathbb{R}^n)$, 若 $f\in\mathcal{S}_{L,\mu}^{1,1}(\mathbb{R}^n)$, 则 $f\in C_L^{1,1}(\mathbb{R}^n)$ 且对任意 $x\in\mathbb{R}^n$ 满足

$$f(x)-f^*\geqslant\frac{\mu}{2}\|x-\bar{x}\|^2. \tag{2.30}$$

文献 [1] 证明了包含关系

$$\mathcal{F}_{L,\mu}^{1,1}(\mathbb{R}^n)\subset\mathcal{W}_{L,\mu}^{1,1}(\mathbb{R}^n)\subset\mathcal{S}_{L,\mu}^{1,1}(\mathbb{R}^n)\subset\mathcal{F}_{L}^{1,1}(\mathbb{R}^n). \tag{2.31}$$

问题模型 2.3 考虑优化问题集合模型 $\mathcal{F}=(\Sigma,\mathcal{O},\mathcal{T}_\epsilon)$:

- 全局信息 Σ: $f\in\mathcal{F}_L^{1,1}(\mathbb{R}^n)$ (或 $f\in\mathcal{F}_{L,\mu}^{1,1}(\mathbb{R}^n)$, 或 $f\in\mathcal{W}_{L,\mu}^{1,1}(\mathbb{R}^n)$, 或 $f\in\mathcal{S}_{L,\mu}^{1,1}(\mathbb{R}^n)$);
- 局部信息 $\mathcal{O}$: $\mathcal{FO}$ 子程序 (或 $\mathcal{PO}$ 子程序);
- 解的精度 $\mathcal{T}_\epsilon$: 求全局极小点 $\bar{x}\in\mathbb{R}^n$, 使得 $f(\bar{x})-f^*\leqslant\epsilon$ (或 $\|\bar{x}-x^*\|\leqslant\epsilon$).

由文献 [1, 11] 可以得到强凸函数集合、弱强凸函数集合和二阶增长函数集合的复杂度上界. 定理 2.3–2.5 给出了这 3 类集合的复杂度分析结论, 其证明可以在文献 [1, 11] 中找到.

定理 2.3 若 $f\in\mathcal{F}_L^{1,1}(\mathbb{R}^n)$, 令步长取 $h_k\equiv h=\frac{2}{L}$, 则梯度法求解问题模型 2.3 的收敛速率为

$$f(x_k)-f^*\leqslant\frac{2L\|x_0-x^*\|^2}{k+4}. \tag{2.32}$$

记 $D_0=\|x_0-x^*\|$, 则复杂度上界为

$$\mathcal{O}\left(\frac{LD_0}{\epsilon}\right). \tag{2.33}$$

注意到梯度法求解该问题模型时, 表现为全局次线性收敛, 且复杂度上界为 $\mathcal{O}(\frac{1}{\epsilon})$. 令 $Q=L/\mu$.

定理 2.4 若 $f\in\mathcal{F}_{L,\mu}^{1,1}(\mathbb{R}^n)$, 令步长取 $h_k\equiv h=\frac{1}{L}$, 则梯度法求解问题模型 2.3 的收敛速率为

$$\|x_k-x^*\|^2\leqslant\left(\frac{Q-1}{Q+1}\right)^{2k}\|x_0-x^*\|^2, \tag{2.34}$$

$$f(x_k)-f^*\leqslant\frac{L}{2}\left(\frac{Q-1}{Q+1}\right)^{2k}\|x_0-x^*\|^2. \tag{2.35}$$

目标函数值对应的复杂度上界为

$$\log\left(\frac{LD_0^2}{\epsilon}\right)\Big/\log\left(\frac{Q-1}{Q+1}\right)^2. \tag{2.36}$$

注意到梯度法求解该问题模型时, 表现为全局线性收敛, 且复杂度上界为 $\mathcal{O}(\ln\frac{1}{\epsilon})$.

定理 2.5 若 $f\in\mathcal{W}_{L,\mu}^{1,1}(\mathbb{R}^n)$, 令步长取 $h_k\equiv h=\frac{1}{L}$, 则梯度法求解问题模型 2.3 的收敛速率为

$$\|x_k-\bar{x}^k\|^2\leqslant\left(\frac{Q-1}{Q+1}\right)^k\|x_0-\bar{x}^k\|^2, \tag{2.37}$$

$$f(x_k)-f^*\leqslant\frac{L}{2}\left(\frac{Q-1}{Q+1}\right)^{k-1}\|x_0-\bar{x}^k\|^2. \tag{2.38}$$

对应的复杂度上界分别为

$$\log\left(\frac{LD_0^2}{\epsilon}\right)\Big/\log\left(\frac{Q-1}{Q+1}\right). \tag{2.39}$$

注意到梯度法求解该问题模型时, 表现为全局线性收敛, 且复杂度上界为 $\mathcal{O}(\ln\frac{1}{\epsilon})$.

定理 2.6 若 $f\in\mathcal{S}_{L,\mu}^{1,1}(\mathbb{R}^n)$, 令步长取 $h_k\equiv h=\frac{1}{L}$, 则梯度法求解问题模型 2.3 的收敛速率为

$$\|x_k-\bar{x}^k\|^2\leqslant\left(\frac{Q}{Q+1}\right)^k\|x_0-\bar{x}^k\|^2, \tag{2.40}$$

$$f(x_k)-f^*\leqslant\frac{L}{2}\left(\frac{Q}{Q+1}\right)^{k-1}\|x_0-\bar{x}^k\|^2. \tag{2.41}$$

对应的复杂度上界分别为

$$\log\left(\frac{LD_0^2}{\epsilon}\right)\Big/\log\left(\frac{Q}{Q+1}\right). \tag{2.42}$$

注意到梯度法求解该问题模型时, 表现为全局线性收敛, 且复杂度上界为 $\mathcal{O}(\ln\frac{1}{\epsilon})$.

比较定理 2.4–2.6, 注意到

$$\log\left(\frac{Q-1}{Q+1}\right)^2\leqslant\log\left(\frac{Q-1}{Q+1}\right)\leqslant\log\left(\frac{Q}{Q+1}\right). \tag{2.43}$$

因此, $\mathcal{F}_{L,\mu}^{1,1}(\mathbb{R}^n)$、$\mathcal{W}_{L,\mu}^{1,1}(\mathbb{R}^n)$ 和 $\mathcal{S}_{L,\mu}^{1,1}(\mathbb{R}^n)$ 的复杂度上界依次递减.

文献 [3] 给出了 Newton 法 (算法 2.2) 的复杂度分析. 首先我们给出 Newton 法的具体形式.

算法 2.2 Newton 法

输入: $x_0\in\mathbb{R}^n$.

对 $k=0,1,\ldots$ 执行迭代,

$$x_{k+1}=x_k-[\nabla^2f(x_k)]^{-1}\nabla f(x_k). \tag{2.44}$$

问题模型 2.4 考虑无约束非凸优化问题集合 $\mathcal{F}=(\Sigma,\mathcal{O},\mathcal{T}_\epsilon)$:

• 全局信息 Σ: 目标函数 $f\in C_L^{2,2}(\mathbb{R}^n)$, 且 $f(x)$ 不一定是凸函数; $f(x)$ 存在局部极小点 x^*, 且 $\nabla f(x^*)$ 正定; 存在常数 $m>0$ 使得 $\nabla^2f(x^*)\succeq mI_n$; 初始点 x_0 距离 x^* 足够近; 约束集合 $X\equiv\mathbb{R}^n$;

• 局部信息 $\mathcal{O}$: 2nd$\mathcal{O}$ 子程序, 返回 $f(x)$ 在 x_0 处的函数值、一阶梯度信息和二阶梯度信息分别为 $f(x_0)$、$\nabla f(x_0)$ 和 $\nabla^2 f(x_0)$;

• 解的精度 $\mathcal{T}_\epsilon$: 求局部极小值的近似解 $\bar{x} \in \mathbb{R}^n$, 使得 $\|\bar{x} - x^*\| \leqslant \epsilon$.

定理 2.7 假设 Newton 法的初始点 x_0 距离局部极小点 x^* 足够近, 满足 $\|x_0 - x^*\| < \bar{r} = \frac{3m}{L}$, 则对任意 k 满足 $\|x_k - x^*\| \leqslant \bar{r}$. Newton 法求解问题模型 2.4 的收敛速率为

$$\|x_{k+1} - x^*\| \leqslant \frac{L\|x_k - x^*\|^2}{2(m - L\|x_k - x^*\|)}. \tag{2.45}$$

复杂度上界为 $c \ln\ln \frac{\gamma}{\epsilon}$, 其中 c 和 γ 为常数.

定理 2.7 的证明可以在文献 [3] 中找到. 注意到 Newton 法求解该问题模型时, 表现为局部二阶收敛, 且复杂度上界为 $\mathcal{O}(\ln\ln \frac{1}{\epsilon})$.

2.3 复杂度下界

Nesterov [3] 给出了光滑凸优化问题和光滑强凸优化问题的复杂度下界.

定理 2.8 (光滑凸优化问题的复杂度下界) 对任意 $x_0 \in \mathbb{R}^n$ 和 $1 \leqslant k \leqslant \frac{1}{2}(n-1)$, 存在 $f \in C_L^{\infty,1}(\mathbb{R}^n)$, 使得对任意 $x_k \in x_0 + \mathrm{span}\{\nabla f(x_0), \ldots, \nabla f(x_{k-1})\}$ 都有

$$f(x_k) - f^* \geqslant \frac{3L\|x_0 - x^*\|^2}{32(k+1)^2}. \tag{2.46}$$

令 $D_0 = \|x_0 - x^*\|$, 令定理 2.8 不等式 (2.46) 右端等于 ϵ, 可以得到一阶算法 (只利用梯度信息的算法) 求解凸优化问题 $C_L^{\infty,1}(\mathbb{R}^n)$ 的复杂度下界为

$$\mathcal{O}\left(\sqrt{\frac{L}{\epsilon}} D_0\right). \tag{2.47}$$

与梯度法的复杂度上界 (2.33) $\mathcal{O}(LD_0/\epsilon)$ 比较, 发现梯度法并不是最优算法.

定理 2.9 (光滑强凸优化问题的复杂度下界) 对任意 $x_0 \in \mathbb{R}^\infty$, $\mu, L > 0$ 和 $Q = L/\mu > 1$, 存在 $f \in C_{L,\mu}^{\infty,1}(\mathbb{R}^\infty)$, 使得对任意 $x_k \in x_0 + \mathrm{span}\{\nabla f(x_0), \ldots, \nabla f(x_{k-1})\}$ 都有

$$f(x_k) - f^* \geqslant \frac{\mu}{2}\left(\frac{\sqrt{Q}-1}{\sqrt{Q}+1}\right)^{2k} \|x_0 - x^*\|^2. \tag{2.48}$$

令不等式 (2.48) 右端等于 ϵ, 利用不等式 $(1+a)^k \leqslant \mathrm{e}^{ka}$ 可以得到一阶算法 (只利用梯度信息的算法) 求解强凸优化问题 $\mathcal{F}_{L,\mu}^{\infty,1}(\mathbb{R}^n)$ 的复杂度下界为

$$\mathcal{O}\left[\sqrt{\frac{L}{\mu}} \max\left(\log \frac{\mu D_0}{\epsilon}, 1\right)\right]. \tag{2.49}$$

2.4 Nesterov 加速梯度法框架

由于现实问题中数据量规模的增加, 优化算法越来越对迭代速度有了要求. 因此, 加速一阶优化算法在近 10 年内受到了非常广泛的关注. 文献 [3, 11–14, 18] 介绍了加速一阶优化算法的发展历史和分析技巧.

2.4.1 凸优化问题

上一节分析了光滑凸优化问题的复杂度下界. 由定理 2.8 知函数集合 $C_L^{\infty,1}(\mathbb{R}^n)$ 相对于解算法集合 $x_k \in x_0 + \mathrm{span}\{\nabla f(x_0), \ldots, \nabla f(x_{k-1})\}$ 的复杂度下界是 $\mathcal{O}(1/\sqrt{\epsilon})$. 又由 (2.33) 知梯度法相对于 $C_L^{\infty,1}(\mathbb{R}^n)$ 的复杂度上界为 $\mathcal{O}(1/\epsilon)$. 因此, 梯度法相对于该解算法集合不是最优的, 我们需要寻找达到函数集合 $C_L^{\infty,1}(\mathbb{R}^n)$ 复杂度下界的最优解算法. Nesterov 在文献 [11] 中提出了可以达到 $\mathcal{O}(1/\sqrt{\epsilon})$ 复杂度的加速梯度法. 为了将 Nesterov 加速梯度法移植到其他新的算法中, 我们需要总结文献 [11] 中加速梯度法的结构, 构造 Nesterov 加速梯度法的一般框架, 并给出加速类优化算法收敛性统一的分析工具和技巧. 首先, 我们给出求解问题 (2.1) 的 Nesterov 加速梯度法框架 (算法 2.3).

算法 2.3 Nesterov 加速梯度算法框架

输入: 令 $x_0 = y_0$, 选取序列 $\{\gamma_k\}$ 和 $\{\beta_k\}$ 使其满足 $L\gamma_k \leqslant \beta_k$, $\gamma_1 = 1$.
对于$k = 1, \ldots, N$ 执行迭代,
1. $z_k = (1-\gamma_k)y_{k-1} + \gamma_k x_{k-1}$;
2. $x_k = \mathrm{argmin}_{x\in X}\{\langle \nabla f(z_k), x\rangle + \frac{\beta_k}{2}\|x - x_{k-1}\|_2^2\}$;
3. $y_k = (1-\gamma_k)y_{k-1} + \gamma_k x_k$.
输出: y_N.

算法 2.3 共有三个迭代序列 $\{x_k\}$、$\{y_k\}$ 和 $\{z_k\}$, 以及两个待定参数序列 $\{\gamma_k\}$ 和 $\{\beta_k\}$, 其中收敛点列为 $\{y_k\}$. 如果约束集合 $X \equiv \mathbb{R}^n$, 则迭代的第二步等价于 $x_k = x_{k-1} - \frac{1}{\beta_k}\nabla f(z_k)$, 此时我们可以画出迭代序列的关系图如图 5 所示.

注意到 y_k、z_k 和 y_{k-1} 组成的三角形与 x_k、x_{k-1} 和 y_{k-1} 组成的三角形相似, 且线段 $\overline{z_k y_{k-1}}$ 与 $\overline{x_{k-1}y_{k-1}}$ 的比为 γ_k.

通过选取不同的参数 γ_k 和 β_k, 我们可以得到加速梯度算法框架不同的收敛速度. 下面给出一般性的分析工具, 通过构造序列 $\{\Gamma_k\}$ 来分析收敛性.

引理 2.1 令 $\gamma_t \in (0,1]$, $t = 1, 2, \ldots$, 构造序列

$$\Gamma_t = \begin{cases} 1, & t = 1, \\ (1-\gamma_t)\Gamma_{t-1}, & t \geqslant 2. \end{cases} \tag{2.50}$$

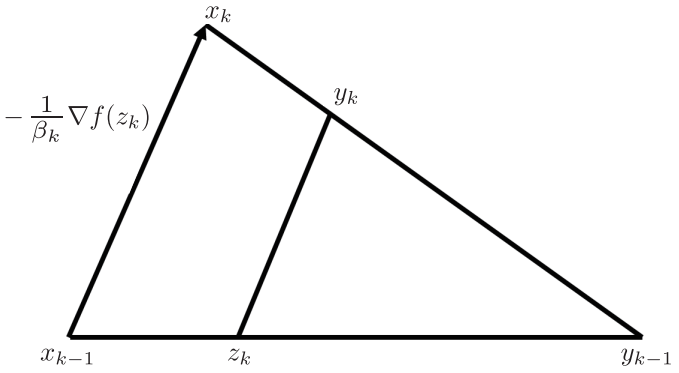

图 5 Nesterov 加速梯度法图

如果序列 $\{\Delta_t\}_{t\geqslant 0}$ 满足

$$\Delta_t \leqslant (1-\gamma_t)\Delta_{t-1} + B_t, \quad t = 1, 2, \ldots, \tag{2.51}$$

则对任意的 k, 对 Δ_k 有估计

$$\Delta_k \leqslant \Gamma_k(1-\gamma_1)\Delta_0 + \Gamma_k \sum_{t=1}^{k} \frac{B_t}{\Gamma_t}. \tag{2.52}$$

在分析算法的收敛速度时, 我们一般根据所要估计的收敛序列, 令 $\Delta_k = f(x_k) - f(x^*)$ 或 $\Delta_k = \|x_k - x^*\|_2^2$, 然后代入构造引理 2.1 中的不等式 (2.51) 得到

$$f(x_k) - f(x^*) \leqslant (1-\gamma_k)(f(x_{k-1}) - f(x^*)) + B_k, \tag{2.53}$$

或者

$$\|x_k - x^*\|_2^2 \leqslant (1-\gamma_k)\|x_{k-1} - x^*\|_2^2 + B_k. \tag{2.54}$$

通常, 我们构造序列 $\{\gamma_t\}_{t\geqslant 1}$ 使其满足 $\gamma_1 = 1$. 根据引理 2.1 不等式 (2.52) 可以得到收敛速度的估计不等式

$$f(x_k) - f(x^*) \leqslant \Gamma_k \sum_{i=1}^{k} \frac{B_i}{\Gamma_i}, \tag{2.55}$$

或者

$$\|x_k - x^*\|_2^2 \leqslant \Gamma_k \sum_{i=1}^{k} \frac{B_i}{\Gamma_i}. \tag{2.56}$$

通过估计上式右端的收敛阶数, 我们可以得到算法的收敛速度. 可以看到收敛速度与 B_k 和 Γ_k 有关. 对于 B_k, 我们需要通过放缩估计其上界; 而对于 Γ_k, 我们可以采取不同的构造方式. 由 (2.50) 可得

$$\Gamma_k = (1-\gamma_k)(1-\gamma_{k-1})\cdots(1-\gamma_2), \quad k = 2, \ldots \tag{2.57}$$

因此在保证 $\gamma_1 = 1$ 的情形下, 我们可以构造序列 $\{\gamma_t\}_{t\geqslant 1}$ 和 $\{\Gamma_t\}_{t\geqslant 1}$, 例如,

$$\gamma_k = \frac{1}{k} \Rightarrow \Gamma_k = \frac{1}{k}, \tag{2.58}$$

$$\gamma_k = \frac{2}{k+1} \Rightarrow \Gamma_k = \frac{2}{k(k+1)}, \tag{2.59}$$

$$\gamma_k = \frac{3}{k+2} \Rightarrow \Gamma_k = \frac{6}{k(k+1)(k+2)}. \tag{2.60}$$

令 $D_X = \sup_{x,y\in X}\|x-y\|$, 下面证明算法 2.3 在取不同 β_k 和 γ_k 序列时的收敛速度.

定理 2.10 若 $f \in C_L^{1,1}(\mathbb{R}^n)$ 且是凸函数, 则 Nesterov 加速梯度算法求解问题模型 2.3 的收敛速率如下.

(1) 取 $\beta_k = L$, $\gamma_k = \frac{1}{k}$, 则 $\Gamma_k = \frac{1}{k}$, $\frac{\beta_k\gamma_k}{\Gamma_k} = L$, 我们有

$$f(y_k) - f(x^*) \leqslant \frac{L}{2k}D_X^2, \quad f(y_k) - f(x^*) \leqslant \frac{L}{2k}\|x_0 - x^*\|^2. \tag{2.61}$$

(2) 取 $\beta_k = \frac{2L}{k}$, $\gamma_k = \frac{2}{k+1}$, 则 $\Gamma_k = \frac{2}{k(k+1)}$, $\frac{\beta_k\gamma_k}{\Gamma_k} = 2L$, 我们有

$$f(y_k) - f(x^*) \leqslant \frac{2L}{k(k+1)}D_X^2, \quad f(y_k) - f(x^*) \leqslant \frac{4L}{k(k+1)}\|x_0 - x^*\|^2. \tag{2.62}$$

(3) 取 $\beta_k = \frac{3L}{k+1}$, $\gamma_k = \frac{3}{k+2}$, 则 $\Gamma_k = \frac{6}{k(k+1)(k+2)}$, $\frac{\beta_k\gamma_k}{\Gamma_k} = \frac{3Lk}{2} \geqslant \frac{\beta_{k-1}\gamma_{k-1}}{\Gamma_{k-1}}$, 我们有

$$f(y_k) - f(x^*) \leqslant \frac{9L}{2(k+1)(k+2)} D_X^2. \tag{2.63}$$

证明 对于算法 2.3, 其收敛点列为 $\{y_k\}$, 如果我们可以证明不等式

$$f(y_k) - f(x^*) \leqslant (1-\gamma_k)[f(y_{k-1}) - f(x^*)] + B_k \tag{2.64}$$

成立. 令 $\Delta_k = f(y_k) - f(x^*)$, 利用引理 2.1 并取 γ_k 满足 $\gamma_0 = 1$, 由 (2.52) 可以得到收敛速度的估计不等式

$$f(y_k) - f(x^*) \leqslant \Gamma_k \sum_{i=1}^{k} \frac{B_i}{\Gamma_i}. \tag{2.65}$$

选择合适的 γ_k 使得上式右端收敛, 我们就可以得到 Nesterov 加速梯度算法的收敛性.

至此, 我们可以用的条件为 $f(x) \in C_L^{1,1}(X)$, $f(x)$ 的凸性, 算法第 1 和 3 步的迭代关系, 算法第 2 步的最优性条件. 首先, 利用 $f(x) \in C_L^{1,1}(X)$, 有

$$f(y_k) \leqslant f(z_k) + \langle \nabla f(z_k), y_k - z_k \rangle + \frac{L}{2}\|y_k - z_k\|^2. \tag{2.66}$$

用迭代步 3 减去步 1 有 $y_k - z_k = \gamma_k(x_k - x_{k-1})$, 将此等式和 $y_k = (1-\gamma_k)y_{k-1} + \gamma_k x_k$ 代入上式右端, 可得

$$f(y_k) \leqslant f(z_k) + \langle \nabla f(z_k), (1-\gamma_k)y_{k-1} + \gamma_k x_k - z_k \rangle + \frac{L\gamma_k^2}{2}\|x_k - x_{k-1}\|^2. \tag{2.67}$$

注意到

$$\begin{aligned} &f(z_k) + \langle \nabla f(z_k), (1-\gamma_k)y_{k-1} + \gamma_k x_k - z_k \rangle \\ &\quad = (1-\gamma_k)[f(z_k) + \langle \nabla f(z_k), y_{k-1} - z_k \rangle] + \gamma_k[f(z_k) + \langle \nabla f(z_k), x_k - z_k \rangle]. \end{aligned} \tag{2.68}$$

由 $f(x)$ 的凸性有

$$f(y_k) \leqslant (1-\gamma_k)f(y_{k-1}) + \gamma_k[f(z_k) + \langle \nabla f(z_k), x_k - z_k \rangle] + \frac{L\gamma_k^2}{2}\|x_k - x_{k-1}\|^2. \tag{2.69}$$

由步 2, $x_k = \operatorname{argmin}_{x\in X}\{\langle \nabla f(z_k), x \rangle + \frac{\beta_k}{2}\|x - x_{k-1}\|_2^2\}$, 其最优性条件为

$$\langle \nabla f(z_k) + \beta_k(x_k - x_{k-1}), x_k - x \rangle \leqslant 0, \quad \forall\, x \in X, \tag{2.70}$$

即

$$\langle x_{k-1} - x_k, x_k - x \rangle \leqslant \frac{1}{\beta_k}\langle \nabla f(x_k), x - x_k \rangle. \tag{2.71}$$

由

$$\begin{aligned} \frac{1}{2}\|x_k - x_{k-1}\|^2 &= \frac{1}{2}\|x_{k-1} - x\|^2 - \langle x_{k-1} - x_k, x_k - x \rangle - \frac{1}{2}\|x_k - x\|^2 \\ &\leqslant \frac{1}{2}\|x_{k-1} - x\|^2 + \frac{1}{\beta_k}\langle \nabla f(z_k), x - x_k \rangle - \frac{1}{2}\|x_k - x\|^2 \end{aligned} \tag{2.72}$$

左右两边同乘 $L\gamma_k^2$, 考虑到 $L\gamma_k \leqslant \beta_k$, 有

$$\frac{L\gamma_k^2}{2}\|x_k - x_{k-1}\|^2 \leqslant \frac{L\gamma_k^2}{2}\|x_{k-1} - x\|^2 + \gamma_k\langle \nabla f(z_k), x - x_k \rangle - \frac{L\gamma_k^2}{2}\|x_k - x\|^2. \tag{2.73}$$

将 (2.69) 与 (2.73) 相加, 利用 $f(x)$ 的凸性 $f(z_k)+\langle\nabla f(z_k),x-z_k\rangle\leqslant f(x)$ 得

$$f(y_k)-f(x)\leqslant(1-\gamma_k)[f(y_{k-1})-f(x)]+\frac{L\gamma_k^2}{2}(\|x_{k-1}-x\|^2-\|x_k-x\|^2). \tag{2.74}$$

令 $\Delta_k=f(y_k)-f(x)$, $B_k=\frac{L\gamma_k^2}{2}(\|x_{k-1}-x\|^2-\|x_k-x\|^2)$, 利用引理 2.1, 有

$$f(y_k)-f(x)\leqslant\frac{\Gamma_k(1-\gamma_1)}{\Gamma_1}(f(y_0)-f(x))+\frac{\Gamma_k}{2}\sum_{i=1}^{k}\frac{\beta_i\gamma_i}{\Gamma_i}(\|x_{i-1}-x\|^2-\|x_i-x\|^2). \tag{2.75}$$

分析上面不等式, 注意到,

$$\begin{aligned}
&\sum_{i=1}^{k}\frac{\beta_i\gamma_i}{\Gamma_i}(\|x_{i-1}-x\|^2-\|x_i-x\|^2)\\
&\quad=\frac{\beta_1\gamma_1}{\Gamma_1}\|x_0-x\|^2+\sum_{i=2}^{k}\left(\frac{\beta_i\gamma_i}{\Gamma_i}-\frac{\beta_{i-1}\gamma_{i-1}}{\Gamma_{i-1}}\right)\|x_{i-1}-x\|^2-\beta_k\gamma_k\Gamma_k\|x_k-x\|^2\\
&\quad\leqslant\frac{\beta_1\gamma_1}{\Gamma_1}\|x_0-x\|^2+\sum_{i=2}^{k}\left(\frac{\beta_i\gamma_i}{\Gamma_i}-\frac{\beta_{i-1}\gamma_{i-1}}{\Gamma_{i-1}}\right)\cdot D_X^2\\
&\quad\leqslant\frac{\beta_k\gamma_k}{\Gamma_k}D_X^2.
\end{aligned}\tag{2.76}$$

因此, 对任意序列 $\{\beta_k\}$、$\{\gamma_k\}$ 和 $\{\Gamma_k\}$ 满足 $L\gamma_k\leqslant\beta_k$ 时, 有

$$f(y_k)-f(x)\leqslant\frac{\beta_k\gamma_k}{2}D_X^2. \tag{2.77}$$

若序列 $\{\beta_k\}$、$\{\gamma_k\}$ 和 $\{\Gamma_k\}$ 满足 $L\gamma_k\leqslant\beta_k$ 且

$$\frac{\beta_k\gamma_k}{\Gamma_k}\geqslant\frac{\beta_{k-1}\gamma_{k-1}}{\Gamma_{k-1}},\quad\text{对任意 } k\geqslant 2, \tag{2.78}$$

则有

$$f(y_k)-f(x)\leqslant\Gamma_k\frac{\beta_1\gamma_1}{2}\|x_0-x\|^2. \tag{2.79}$$

综上, 我们可以通过构造序列 $\{\beta_k\}$ 和 $\{\gamma_k\}$ 来得到算法的收敛速度, 将 β_k、γ_k 和 Γ_k 分别代入不等式 (2.77) 和 (2.79) 即得收敛性. □

2.4.2 非凸优化问题

上一节介绍了 Nesterov 加速梯度算法框架在凸函数上的收敛性质. 对于非凸函数, 如果直接利用 Nesterov 加速梯度算法框架, 不一定有收敛性结果. 对于非凸函数, 我们衡量 $\min_{0\leqslant k\leqslant N}\|\nabla f(x_k)\|$ 的收敛速度. 由 (2.1), 我们知道梯度算法对于非凸和凸情形下的函数集合 $C_L^{1,1}(\mathbb{R}^n)$, 近似解 x_k 满足

$$\min_{0\leqslant k\leqslant N}\|\nabla f(x_k)\|^2\leqslant\frac{2L(f(x_0)-f^*)}{N+1}. \tag{2.80}$$

文献 [15] 给出了 Nesterov 加速梯度算法框架的变形, 从而得到求解无约束非凸优化问题的收敛性. 首先, 我们给出非凸优化问题模型.

问题模型 2.5 考虑无约束非凸优化问题集合 $\mathcal{F}=(\Sigma,\mathcal{O},\mathcal{T}_\epsilon)$:

- 全局信息 Σ: 目标函数 $f\in C_L^{1,1}(\mathbb{R}^n)$, 可能为非凸函数; 约束集合 $X\equiv\mathbb{R}^n$;
- 局部信息 $\mathcal{O}$: $\mathcal{FO}$ 子程序;
- 解的精度 $\mathcal{T}_\epsilon$: 求局部极小点 $\bar{x}\in\mathbb{R}^n$, 使得 $\|\nabla f(\bar{x})\|\leqslant\epsilon$.

我们对 Nesterov 加速梯度算法框架做一些变形, 加入新的参数序列 λ_k 并修改 y_k 的更新方式, 得到非凸函数的加速梯度法 (算法 2.4).

算法 2.4 非凸函数的加速梯度法

输入: 令 $x_0=y_0$, $\gamma_k\in(0,1]$.
对于$k=1,\ldots,N$ 执行迭代,
1. $z_k=(1-\gamma_k)y_{k-1}+\gamma_k x_{k-1}$;
2. $x_k=x_{k-1}-\frac{1}{\beta_k}\nabla f(z_k)$;
3. $y_k=z_k-\frac{1}{\lambda_k}\nabla f(z_k)$.
输出: z_N.

定理 2.11 取 $\gamma_k=\frac{2}{k+1}$ 和 $\lambda_k=2L$, 利用算法 2.4 求解问题模型 2.5 有如下收敛性结果:
(1) 若取 $\beta_k\in[\frac{4k+4}{2k+3}L,2L]$, 则

$$\min_{0\leqslant k\leqslant N}\|\nabla f(z_k)\|^2\leqslant\frac{6L(f(x_0)-f^*)}{N}. \tag{2.81}$$

(2) 如果问题模型 2.5 中 $f(x)$ 是凸函数, 取 $\beta_k=\frac{4L}{k}$, 则

$$\min_{0\leqslant k\leqslant N}\|\nabla f(z_k)\|^2\leqslant\frac{96L^2\|x_0-x^*\|^2}{N(N+1)(N+2)}. \tag{2.82}$$

该定理的证明可以在文献 [15] 中找到. 注意到, 在凸函数情形下, 算法 2.4 迭代点梯度 $\|\nabla f(z_k)\|$ 的收敛速度为 $\mathcal{O}(1/N^{3/2})$, 而梯度法的收敛速度为 $\mathcal{O}(1/N^{1/2})$. 而在非凸函数情形下, 算法 2.4 与梯度法的收敛速度相同.

3 非光滑优化问题的复杂度分析

本节研究非光滑优化问题的复杂度分析. 我们介绍三部分内容: 次梯度法和镜像梯度法的收敛性和复杂度, 求解非光滑优化问题的重心法和椭球法, 复合优化问题的加速算法. 次梯度法和镜像梯度法可以参见文献 [2,3,12], 非光滑优化问题的复杂度下界可以参见文献 [3,12], 重心法和椭球法可以参见文献 [5,19,20], 复合优化问题可参见文献 [3,12–15,17,21–23].

3.1 次梯度法与镜像梯度法

考虑优化问题

$$f^*=\min_{x\in X}f(x). \tag{3.1}$$

当目标函数光滑且无约束时, 对给定初始点 $x_0\in\mathbb{R}^n$, 梯度法根据如下迭代公式更新迭代点 x_k:

$$x_{k+1}=x_k-\gamma_k\nabla f(x_k), \tag{3.2}$$

其中 $\gamma_k > 0$ 是第 k 步迭代的步长. 为了求解非光滑有约束的优化问题 (3.1), 我们需要对梯度法 (3.2) 做两个改变. 第一, 由于目标函数 f 不一定可导, 我们需要将导数 $\nabla f(x_k)$ 用一个次梯度 $g_k \in \partial f(x_k)$ 替换. 第二, 由于迭代公式 (3.2) 只适用于无约束情形, 对于有约束问题 $X \neq \mathbb{R}^n$, (3.2) 确定的新迭代点 x_{k+1} 有可能落在约束集合 X 外面, 因此需要将 x_{k+1} 投影回约束集合 X. 我们得到非光滑约束优化问题的投影次梯度算法 (参见算法 3.1).

算法 3.1 投影次梯度法

输入: $x_0 \in \mathbb{R}^n$.

对 $k = 0, 1, \ldots$ 执行迭代,

$$x_{k+1} = \operatorname*{argmin}_{x \in X} \|x - (x_k - \gamma_k g(x_k))\|_2, \tag{3.3}$$

其中 $\gamma_k > 0$ 且 $g(x_k) \in \partial f(x_k)$.

我们可以用对目标函数做近似逼近的角度来理解投影次梯度法的迭代 (3.3). 事实上, (3.3) 可以被写成

$$\begin{aligned} x_{k+1} &= \operatorname*{argmin}_{x \in X} \frac{1}{2}\|x - (x_k - \gamma_k g(x_k))\|_2^2 \\ &= \operatorname*{argmin}_{x \in X} \gamma_k \langle g(x_k), x - x_k\rangle + \frac{1}{2}\|x - x_k\|_2^2 \\ &= \operatorname*{argmin}_{x \in X} \gamma_k \langle g(x_k), x\rangle + \frac{1}{2}\|x - x_k\|_2^2 \\ &= \operatorname*{argmin}_{x \in X} f(x_k) + \langle g(x_k), x - x_k\rangle + \frac{1}{2\gamma_k}\|x - x_k\|_2^2. \end{aligned} \tag{3.4}$$

这意味着我们需要最小化 $f(x)$ 在 X 上的线性近似 $f(x_k) + \langle g(x_k), x - x_k\rangle$, 同时求得的解 x_{k+1} 离上一步迭代点 x_k 不能太远 (被 $\frac{1}{2\gamma_k}\|x - x_k\|_2^2$ 控制). 为了得到投影梯度法的收敛速率, 我们需要对 x_{k+1} 的性质做一定的刻画, 有下述引理.

引理 3.1 投影次梯度法 (3.3) 确定的 x_{k+1}, 对任意 $x \in X$, 有

$$\gamma_k \langle g(x_k), x_{k+1} - x\rangle + \frac{1}{2}\|x_{k+1} - x_k\|_2^2 \leqslant \frac{1}{2}\|x - x_k\|_2^2 + \frac{1}{2}\|x - x_{k+1}\|_2^2. \tag{3.5}$$

证明 记 $\phi(x) = \gamma_k \langle g(x_k), x\rangle + \frac{1}{2}\|x - x_k\|_2^2$, 由 $\phi(x)$ 的强凸性, 有

$$\phi(x) \geqslant \phi(x_{k+1}) + \langle \phi'(x_{k+1}), x - x_{k+1}\rangle + \frac{1}{2}\|x - x_{k+1}\|_2^2. \tag{3.6}$$

注意到 $x_{k+1} = \operatorname{argmin}_{x \in X} \phi(x)$, 由其一阶最优性条件, 有

$$\langle \phi'(x_{k+1}), x - x_{k+1}\rangle \geqslant 0, \quad \text{对任意 } x \in X. \tag{3.7}$$

因此,

$$\phi(x) - \phi(x_{k+1}) \geqslant \frac{1}{2}\|x - x_{k+1}\|_2^2. \tag{3.8}$$

整理上式即可得 (3.5). □

考虑 $X \subset \mathbb{R}^n$ 闭且凸, $f : X \to \mathbb{R}^n$ 连续且凸, 其次梯度满足

$$\|g(x)\| \leqslant M \tag{3.9}$$

对任意 $x \in X$ 成立.

定理 3.1 令 x_k $(k=1,\dots,N)$ 由 (3.3) 产生, 且定义

$$\bar{x}_s^N=\frac{\gamma_s x_s+\cdots+\gamma_N x_N}{\gamma_s+\cdots+\gamma_N}=\left(\sum_{k=s}^{N}\gamma_k\right)^{-1}\sum_{k=s}^{k}\gamma_k x_k, \tag{3.10}$$

则有

$$f(\bar{x}_s^N)-f^*\leqslant\left(2\sum_{k=s}^{N}\gamma_k\right)^{-1}\left[\|x_s-x^*\|_2^2+M^2\sum_{k=s}^{N}\gamma_k^2\right]. \tag{3.11}$$

证明 由 f 的凸性, 以及引理 3.1, 有

$$\begin{aligned}\gamma_k[f(x_k)-f(x)]&\leqslant\gamma_k\langle g(x_k),x_k-x\rangle\\&=\gamma_k\langle g(x_k),x_k-x_{k+1}\rangle+\gamma_k\langle g(x_k),x_{k+1}-x\rangle\\&\leqslant\gamma_k\langle g(x_k),x_k-x_{k+1}\rangle-\frac{1}{2}\|x_{k+1}-x_k\|_2^2+\frac{1}{2}\|x-x_k\|_2^2-\frac{1}{2}\|x-x_{k+1}\|_2^2\\&\leqslant\frac{\gamma_k^2}{2}\|g(x_k)\|_2^2+\frac{1}{2}\|x-x_k\|_2^2-\frac{1}{2}\|x-x_{k+1}\|_2^2\\&\leqslant\frac{M^2}{2}\|g(x_k)\|_2^2+\frac{1}{2}\|x-x_k\|_2^2-\frac{1}{2}\|x-x_{k+1}\|_2^2,\end{aligned} \tag{3.12}$$

其中不等式 (3.12) 成立是因为 $bc^2+ac^2/2\leqslant b^2/(2a)$. 将上式对 $k=s$ 到 N 求和, 并利用 f 的凸性, 我们可以得到不等式 (3.11). □

推论 3.1 对于算法 3.1, 选取不同的步长策略, 有如下收敛性结果.

(1) 固定步长策略: 假设算法 3.1 的总迭代步数 N 固定, 令 $D_X=\max_{x_1,x_2\in X}\|x_1-x_2\|$, 取固定步长,

$$\gamma_k=\sqrt{\frac{D_X^2}{NM^2}},\quad k=1,\dots,N, \tag{3.13}$$

则

$$f(\bar{x}_1^N)-f^*\leqslant\frac{MD_X}{2\sqrt{N}},\quad \text{对任意 } N\geqslant 1. \tag{3.14}$$

(2) 变步长策略: 取变步长,

$$\gamma_k=\sqrt{\frac{D_X^2}{kM^2}},\quad k=1,\dots, \tag{3.15}$$

则

$$f(\bar{x}_{\lceil k/2\rceil}^N)-f^*\leqslant\mathcal{O}(1)\frac{MD_X}{\sqrt{k}}, \tag{3.16}$$

其中 $\mathcal{O}(1)$ 是固定常数.

证明 当固定步长时, 令 $\gamma_k=\gamma$, $k=1,\dots,N$, 代入定理 3.1 得不等式

$$f(\bar{x}_1^N)-f^*\leqslant\frac{1}{2}\left[\frac{D_X^2}{N\gamma}+M^2\gamma\right].$$

对上式右端求极小, 可得所求的固定步长 γ 取值. 对变步长策略, 只需代入验证即可. □

考虑到算法 3.1 与 $\mathbb{R}^n$ 的结构有关. 更确切地说, 其中的投影步的距离度量选择的是 Euclid 范数 (即 ℓ_2 范数), 同时 D_X 与 M 的值也是在 ℓ_2 范数下计算的. 这就提醒我们可以对投影次梯度法做一定的推广, 使得其在非 Euclid 结构下执行, 这就是镜像梯度法的由来.

令 $\|\cdot\|$ 是 $\mathbb{R}^n$ 上的一个范数, $\|\xi\|_* = \max\{\langle\xi, x\rangle : \|x\| \leqslant 1\}$ 是它的对偶范数. 定义相对于该范数 $\|\cdot\|$ 的距离生成函数 $\omega : X \to \mathbb{R}$, $\omega(x)$ 连续可微且强凸 (相对于范数 $\|\cdot\|$ 和参数 μ), 即

$$\langle\nabla\omega(x) - \nabla\omega(y), x - y\rangle \geqslant \mu\|x - y\|^2, \quad \text{对任意 } x, y \in X. \tag{3.17}$$

最简单的距离生成函数是 $\omega(x) = \|x\|_2^2/2$ (取 ℓ_2 范数且 $\mu = 1$). 有了距离生成函数, 我们可以定义 Bregman 距离,

$$V(x, y) = \omega(y) - [\omega(x) + \langle\nabla\omega(x), y - x\rangle]. \tag{3.18}$$

注意到 $V(x, \cdot)$ 是强凸函数 (相对于范数 $\|\cdot\|$ 和参数 μ). 当取 $\omega(x) = \|x\|_2^2/2$ 时, $V(x, y) = \|y - x\|_2^2/2$.

注意到投影次梯度法 (算法 3.1) 等价于 (3.4), 利用 Bregman 距离, 我们可以构造在新的范数 $\|\cdot\|$ 下的投影次梯度法 (即镜像梯度法, 见算法 3.2).

算法 3.2 镜像梯度法

输入: $x_0 \in \mathbb{R}^n$.

对 $k = 0, 1, \ldots$ 执行迭代,

$$x_{k+1} = \operatorname*{argmin}_{x\in X} \gamma_k\langle g(x_k), x\rangle + V(x_k, x), \tag{3.19}$$

其中 $\gamma_k > 0$ 且 $g(x_k) \in \partial f(x_k)$.

注意到当算法 3.2 中 $V(x, y) = \|y - x\|_2^2/2$ 时, 算法 3.2 就是算法 3.1. 与引理 3.1 类似, 我们也有对镜像梯度法 x_{k+1} 性质的刻画, 有下述引理.

引理 3.2 镜像梯度法确定的 x_{k+1}, 对任意 $x \in X$, 有

$$\gamma_k\langle g(x_k), x_{k+1} - x\rangle + V(x_k, x_{k+1}) \leqslant V(x_k, x) - V(x_{k+1}, x). \tag{3.20}$$

证明 由 (3.19) 的最优性条件, 对任意 $x \in X$, 不等式

$$\langle\gamma_k g(x_k) + \nabla V(x_k, x_{k+1}), x - x_k\rangle \geqslant 0 \tag{3.21}$$

成立, 其中 $\nabla V(x_k, x_{k+1})$ 表示 $V(x_k, \cdot)$ 在 x_{k+1} 处的导数. 由 $V(x, y)$ 的定义 (3.18), 对任意 $x \in X$, 等式

$$V(x_k, x) = V(x_k, x_{k+1}) + \langle\nabla V(x_k, x_{k+1}), x - x_{k+1}\rangle + V(x_{k+1}, x) \tag{3.22}$$

成立. 将 (3.19) 代入上面等式即得结果. □

定理 3.2 令 x_k $(k = 1, \ldots, N)$ 由 (3.19) 产生, 且定义

$$\bar{x}_s^N = \frac{\gamma_s x_s + \cdots + \gamma_N x_N}{\gamma_s + \cdots + \gamma_N} = \left(\sum_{k=s}^N \gamma_k\right)^{-1} \sum_{k=s}^k \gamma_k x_k, \tag{3.23}$$

则有

$$f(\bar{x}_s^N) - f^* \leqslant \left(\sum_{k=s}^N \gamma_k\right)^{-1} \left[V(x_s, x^*) + \frac{M^2}{2\mu}\sum_{k=s}^N \gamma_k^2\right]. \tag{3.24}$$

仿照定理 3.1 的证明方式, 利用引理 3.2 可以证明定理 3.2.

推论 3.2 对于算法 3.2, 选取不同的步长策略, 令 $D_{\omega,X}^2 = \max_{x_1,x_2\in X} V(x_1,x_2)$, 取步长

$$\gamma_k = \sqrt{\frac{2\mu D_{\omega,X}^2}{kM^2}}, \quad k = 1,\dots, \tag{3.25}$$

则

$$f(\bar{x}_1^k) - f^* \leqslant \frac{\sqrt{2}MD_{\omega,X}}{\sqrt{\mu k}}, \quad \text{对任意 } k \geqslant 1. \tag{3.26}$$

证明 将 (3.25) 代入 (3.24) 即得 (3.26) 结果. □

3.2 重心法与椭球法

本小节列出重心法 (算法 3.3) 和椭球法的收敛性和复杂度, 其收敛性证明可以在文献 [5] 中找到.

算法 3.3 重心法

令 $\mathcal{S}_1 = X$, 对 $k = 1,\dots,N$ 执行迭代.

1. 计算集合 $\mathcal{S}_k$ 的重心

$$c_k = \frac{1}{\mathrm{vol}(\mathcal{S}_k)}\int_{x\in\mathcal{S}_k} x dx; \tag{3.27}$$

2. 在 c_k 处调用 $\mathcal{FO}$ 子程序得 $w_k \in \partial f(c_k)$, 更新集合 $\mathcal{S}_{k+1}$,

$$\mathcal{S}_{k+1} = \mathcal{S}_k \cap \{x \in \mathbb{R}^n : (x - c_k)^{\mathrm{T}} w_k \leqslant 0\}, \tag{3.28}$$

输出: $x_N \in \mathrm{argmin}_{1\leqslant r\leqslant N}\ f(c_r)$.

问题模型 3.1 考虑约束凸优化问题集合 $\mathcal{F} = (\Sigma, \mathcal{O}, \mathcal{T}_\epsilon)$:

- 全局信息 Σ: 目标函数 $f(x) \in C(X)$ 且是凸函数, 存在常数 $B > 0$ 使得对任意 $x \in X$ 有 $-B \leqslant f(x) \leqslant B$; 约束集合 $X \subset \mathbb{R}^n$ 闭且凸;
- 局部信息 $\mathcal{O}$: $\mathcal{FO}$ 子程序, 对于任意给定的 x_0 返回函数值 $f(x_0)$ 和次梯度 $\partial f(x_0)$;
- 解的精度 $\mathcal{T}_\epsilon$: 求全局极小点 $\bar{x} \in \mathbb{R}^n$, 使得 $f(\bar{x}) - f^* \leqslant \epsilon$.

引理 3.3 设 $\mathcal{S}$ 是重心在原点的凸集合, 即 $\int_{x\in\mathcal{S}} x dx = 0$, $\mathrm{vol}(\mathcal{S})$ 表示 $\mathcal{S}$ 的体积, 则对任意 $w \in \mathbb{R}^n$, $w \neq 0$, 有

$$\mathrm{vol}(\mathcal{S} \cap \{x \in \mathbb{R}^n : x^{\mathrm{T}} w \geqslant 0\}) \geqslant \frac{1}{\mathrm{e}}\mathrm{vol}(\mathcal{S}). \tag{3.29}$$

定理 3.3 重心法 3.3 在求解问题模型 3.1 时的收敛速率为

$$f(x_N) - f^* \leqslant 2B\left(1 - \frac{1}{\mathrm{e}}\right)^{N/n}. \tag{3.30}$$

复杂度上界为

$$\mathcal{O}\left(n \log \frac{2B}{\epsilon}\right). \tag{3.31}$$

从计算角度来说, 由于每一步都需要计算集合的体积, 即进行积分计算, 因此, 重心法计算量过大, 往往很难执行. 重心法只是提供一个理论上的可行策略. 而椭球法借用了重心法的思想, 改进了其计算可行性.

椭球是具有如下形式的凸集合:

$$\mathcal{E} = \{x \in \mathbb{R}^n : (x - c)^{\mathrm{T}} H^{-1} (x - c) \leqslant 1\}, \tag{3.32}$$

其中 $c\in\mathbb{R}^n$, H 是一个对称正定矩阵. 几何上 c 是椭球 $\mathcal{E}$ 的重心, 而 H 的特征向量是椭球 $\mathcal{E}$ 的半轴, 半轴的长度是对应特征值的正平方根.

引理 3.4 令 $\mathcal{E}_0=\{x\in\mathbb{R}^n:\ (x-c_0)^{\mathrm{T}}H_0^{-1}(x-c_0)\leqslant 1\}$, 对任意 $w\in\mathbb{R}^n$, $\omega\neq 0$, 可以构造椭球 $\mathcal{E}$ 使得

$$\mathcal{E}\supset\{x\in\mathcal{E}_0:\ w^{\mathrm{T}}(x-c_0)\leqslant 0\},\tag{3.33}$$

且原椭球 $\mathcal{E}_0$ 与新椭球 $\mathcal{E}$ 之间体积有如下关系:

$$\mathrm{vol}(\mathcal{E})\leqslant\exp\left(-\frac{1}{2n}\right)\mathrm{vol}(\mathcal{E}_0).\tag{3.34}$$

当 $n\geqslant 2$ 时, 新椭球 $\mathcal{E}=\{x\in\mathbb{R}^n:\ (x-c)^{\mathrm{T}}H^{-1}(x-c)\leqslant 1\}$, 其中

$$c=c_0-\frac{1}{n+1}\frac{H_0w}{\sqrt{w^{\mathrm{T}}H_0w}},\tag{3.35}$$

$$H=\frac{n^2}{n^2-1}\left(H_0-\frac{2}{n+1}\frac{H_0ww^{\mathrm{T}}H_0)}{w^{\mathrm{T}}H_0w}\right).\tag{3.36}$$

问题模型 3.2 考虑约束凸优化问题集合 $\mathcal{F}=(\Sigma,\mathcal{O},\mathcal{T}_\epsilon)$:

• 全局信息 Σ: 目标函数 $f(x)\in C(X)$ 且是凸函数, 存在常数 $B>0$ 使得对任意 $x\in X$ 有 $-B\leqslant f(x)\leqslant B$; 约束集合 $X\subset\mathbb{R}^n$ 闭且凸, 且 $\mathcal{B}(r)\subset X\subset\mathcal{B}(R)$.

• 局部信息 $\mathcal{O}$: $\mathcal{FO}$ 子程序, 对于任意给定的 x_0 返回函数值 $f(x_0)$ 和次梯度 $\partial f(x_0)$; $\mathcal{SO}$ 子程序, 返回分割平面垂直向量 ω 使得

$$w^{\mathrm{T}}(x-c_0)\leqslant 0,\quad\forall\, x\in X.\tag{3.37}$$

• 解的精度 $\mathcal{T}_\epsilon$: 求全局极小点 $\bar{x}\in\mathbb{R}^n$, 使得 $f(\bar{x})-f^*\leqslant\epsilon$.

我们可以估计椭球法 (算法 3.4) 的如下收敛速率和复杂度.

算法 3.4 椭球法 (ellipsoid method)

输入: 令 $\mathcal{E}_0$ 为包含约束集合 X 且半径为 R 中心为 c_0 的球, 令 $H_0=R^2I_n$.

对 $k=1,\ldots,N$ 执行迭代,

1. 若 $c_k\notin X$, 则调用子程序返回向量 $w_k\in\mathbb{R}^n$, 过 c_k 且垂直 w_k 的平面将椭球 $\mathcal{E}_k$ 分割, 使得约束集 $X\subset\{x:(x-c_k)^{\mathrm{T}}w_k\leqslant 0\}$; 若 $c_k\in X$, 则调用子程序返回向量 $w_k\in\partial f(c_k)$.

2. 构造新的椭球 $\mathcal{E}_{k+1}=\{x:\ (x-c_{k+1})^{\mathrm{T}}H_{k+1}^{-1}(x-c_{k+1})\leqslant 1\}$ 使得

$$\{x\in\mathcal{E}_k:\ (x-c_k)^{\mathrm{T}}w_k\leqslant 0\}\subset\mathcal{E}_{k+1},\tag{3.38}$$

其中 c_{k+1} 和 H_{k+1} 满足

$$c_{k+1}=c_k-\frac{1}{n+1}\frac{H_kw}{\sqrt{w^{\mathrm{T}}H_kw}},\tag{3.39}$$

$$H_{k+1}=\frac{n^2}{n^2-1}\left(H_k-\frac{2}{n+1}\frac{H_kww^{\mathrm{T}}H_k)}{w^{\mathrm{T}}H_kw}\right).\tag{3.40}$$

输出: 若 $\{c_1,\ldots,c_N\}\cap X\neq\emptyset$, 则输出

$$x_N\in\operatorname*{argmin}_{c\in\{c_1,\ldots,c_N\}\cap X}f(c).\tag{3.41}$$

定理 3.4 算法 3.4 在求解问题模型 3.2 时的收敛速率为

$$f(x_N)-f^*\leqslant\frac{2BR}{r}\exp\left(-\frac{N}{2n^2}\right).\tag{3.42}$$

复杂度上界为

$$\mathcal{O}\left(n^2 \log \frac{BR}{r\epsilon}\right). \tag{3.43}$$

从上面的定理可以看出,

(1) 从子程序调用的分析复杂度看, 椭球法要比重心法差, 前者需调用 $\mathcal{FO}$ 子程序 $\mathcal{O}(n^2 \log(\frac{2BR}{r\epsilon}))$ 次, 而后者仅需要 $\mathcal{O}(n \log(\frac{2B}{\epsilon}))$ 次;

(2) 从计算角度来看, 椭球法要比重心法更容易执行.

3.3 复合优化的 Nesterov 加速算法

本小节介绍复合优化问题的复杂度分析. 考虑复合优化问题

$$\min_{x\in\mathbb{R}^n} \Phi(x) := \phi(x) + h(x), \quad \phi(x) = f(x) + g(x), \tag{3.44}$$

其中 $f(x) \in C^{1,1}_{L_f}(\mathbb{R}^n)$ 是非凸函数, $g(x) \in C^{1,1}_{L_g}(\mathbb{R}^n)$ 是凸函数, 于是有 $\phi(x) \in C^{1,1}_{L_\phi}$, 其中 $L_\phi = L_f + L_g$. $h(x)$ 是定义域有界的简单凸函数且有可能不光滑 (例如, $h(x) = \mathcal{I}_X(\cdot)$, 其中 $\mathcal{I}_X(\cdot)$ 是凸紧集合 X 的示性函数; $h(x) = \|x\|_1$, 是机器学习损失函数中常用的 ℓ_1 正则项). 可以看到优化问题 (3.44) 由光滑部分 $\phi(x)$ 和非光滑部分 $h(x)$ 组成, 其中光滑部分又分为非凸函数 $f(x)$ 和凸函数 $g(x)$. 本小节介绍复合优化问题的复杂度分析. 当 $f(x) = 0$ 时, 目标函数 (3.44) 是凸函数, 我们使用近似点梯度法 (proximal gradient method) 和加速梯度法 (快速迭代阈值收缩算法 (fast iterative shrinkage-thresholding algorithm, FISTA) 和 Nesterov 加速算法) 求解, 具体可参见文献 [3, 12–14, 21, 22]. 当 $f(x) \neq 0$ 时, 目标函数 (3.44) 是非凸函数, 文献 [15, 17, 23] 给出了非凸函数情形下的复杂度分析.

3.3.1 凸复合优化问题

当 $f(x) = 0$ 时, 复合优化问题 (3.44) 是一个凸优化问题

$$\min_{x\in\mathbb{R}^n} \Phi(x) := g(x) + h(x). \tag{3.45}$$

令 $\mathbf{prox}_h(\cdot)$ 是函数 $h(x)$ 的近似点算子. 对任意 $x \in \mathbb{R}^n$, 函数 $h(x)$ 的近似点 $\mathbf{prox}_h(x)$ 定义为

$$\mathbf{prox}_h(x) = \operatorname*{argmin}_u \left(h(u) + \frac{1}{2}\|u - x\|_2^2\right). \tag{3.46}$$

当 $h(x)$ 是闭凸函数时, $\mathbf{prox}_h(x)$ 存在且唯一. 我们可以用如下近似点梯度法 (算法 3.5) 求解问题 (3.45).

算法 3.5 近似点梯度法

输入: $x_0 \in \mathbb{R}^n$, 步长序列 $\{t_k\}$.

对 $k = 1, 2, \ldots$ 执行迭代,

$$x_k = \mathbf{prox}_{t_k h}(x_{k-1} - t_k \nabla g(x_{k-1})). \tag{3.47}$$

当 $h(x) = 0$ 时, $\mathbf{prox}_h(x) = x$, 算法 3.5 等价于光滑函数的梯度法. 当 $h(x) = \mathcal{I}_C(x)$ 时, $\mathbf{prox}_h(x) = \operatorname{argmin}_{u\in C} \|u - x\|^2$, 算法 3.5 等价于光滑函数的投影梯度法. 当 $h(x) = \|x\|_1$ 时, $\mathbf{prox}_h(x)$ 被称为软阈算子 (soft-threshold operator), 算法 3.5 可以用来求解机器学习中带 ℓ_1 正则的目标优化函数.

定理 3.5 算法 3.5 求解凸复合优化问题 (3.45) 时, 若取固定步长 $t_k = \frac{1}{L_g}$, 则

$$\Phi(x_k) - \Phi(x_*) \leqslant \frac{L_g}{2k}\|x_0 - x_*\|^2. \tag{3.48}$$

可以看到, 算法 3.5 的复杂度为 $\mathcal{O}(\frac{1}{\epsilon})$. 文献 [14] 给出了如下加速版本的近似点梯度法 FISTA (算法 3.6).

算法 3.6 FISTA

输入: $x_0 = x_{-1} \in \mathbb{R}^n$.
对 $k = 1, 2, \ldots$ 执行迭代,

$$y = x_{k-1} + \frac{k-2}{k+1}(x_{k-1} - x_{k-2}), \tag{3.49}$$

$$x_k = \mathbf{prox}_{t_k h}(y - t_k \nabla g(y)). \tag{3.50}$$

下面定理给出了 FISTA 算法的复杂度结果.

定理 3.6 算法 3.6 求解凸复合优化问题 (3.45) 时, 若取固定步长 $t_k = \frac{1}{L_g}$, 则

$$\Phi(x_k) - \Phi(x_*) \leqslant \frac{2L_g}{(k+1)^2}\|x_0 - x_*\|^2. \tag{3.51}$$

复合优化问题 (3.45) 还有另一种加速算法形式 (算法 3.7).

算法 3.7 复合函数的 Nesterov 加速算法

输入: 令 $x_0 = y_0$, 选取序列 $\{\gamma_k\}$ 和 $\{\beta_k\}$ 使其满足 $\gamma_k \leqslant L_g\beta_k$, $\gamma_1 = 1$.
对于$k = 1, \ldots, N$ 执行迭代,
1. $z_k = (1 - \gamma_k)y_{k-1} + \gamma_k x_{k-1}$;
2. $x_k = \mathbf{prox}_{(\beta_k/\gamma_k)h}(x_{k-1} - \frac{\beta_k}{\gamma_k}\nabla g(z_k))$;
3. $y_k = (1 - \gamma_k)y_{k-1} + \gamma_k x_k$.
输出: y_N.

同样, 该算法也有 $\mathcal{O}(\frac{1}{\sqrt{\epsilon}})$ 复杂度结果.

定理 3.7 算法 3.7 求解凸复合优化问题 (3.45) 时, 取 $\gamma_k = \frac{2}{k+1}$, $\beta_k = \frac{1}{L_g}$, 则

$$\Phi(x_k) - \Phi(x_*) \leqslant \frac{2L_g}{(k+1)^2}\|x_0 - x_*\|^2. \tag{3.52}$$

3.3.2 非凸复合优化问题

当 $f(x) \neq 0$ 时, 复合优化问题 (3.44) 是一个非凸优化问题. 在非凸函数设定下, 我们无法用函数值与最优值的差来衡量优化算法解的精度, 因为 $f(x_k) - f(x_*)$ 并非恒大于 0. 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 (3.44), 利用梯度映射 (gradient mapping) 作为停止准则, 其定义如下:

$$G_\alpha(x, y) = \frac{1}{\alpha}[x - \mathbf{prox}_{\alpha h}(x - \alpha y)]. \tag{3.53}$$

由近似点梯度法 (算法 3.5) 可以看到 $G_{t_k}(x_{k-1})$ 是近似点迭代步的负方向, 即

$$x_k - x_{k-1} = \mathbf{prox}_{t_k h}(x_{k-1} - t_k \nabla g(x_{k-1})) - x_{k-1}$$

$$= -t_k G_{t_k}(x_{k-1}, \nabla g(x_{k-1})). \tag{3.54}$$

同时, 由近似点算子次梯度的定义, 有

$$G_\alpha(x, \nabla\phi(x)) \in \nabla\phi(x) + \partial h(x - \alpha G_\alpha(x, \nabla\phi(x))). \tag{3.55}$$

由此可以得到结论: $G_\alpha(x, \nabla\phi(x)) = 0$ 当且仅当 x 是 $\phi(x) + h(x)$ 的局部极小点. 将算法 2.4 做一些修改, 文献 [15] 给出了非凸复合函数的加速梯度法框架, 并用梯度映射作为停止条件分析了算法的复杂度.

我们利用 $\|G_{\beta_k}(z_k, \nabla\phi(z_k))\|$ 作为算法 3.8 的停止准则. 注意到

$$G_{\beta_k}(z_k, \nabla\phi(z_k)) = \frac{1}{\beta_k}(z_k - y_k), \tag{3.56}$$

该停止条件实际上等价于解序列 $\{z_k\}$ 与 $\{x_k\}$ 之间的距离.

算法 3.8 复合函数的加速算法框架

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 $\gamma_k \in (0, 1)$.
对 $k = 1, \ldots, N$, 执行迭代,
1. $z_k = (1 - \gamma_k)y_{k-1} + \gamma_k x_{k-1}$;
2. $x_k = \mathbf{prox}_{\lambda_k h}(x_{k-1} - \lambda_k \nabla\phi(z_k))$;
3. $y_k = \mathbf{prox}_{\beta_k h}(z_k - \beta_k \nabla\phi(z_k))$.
输出: z_N.

定理 3.8 假设对任意 $\alpha \in (0, +\infty)$ 和任意 $x, y \in \mathbb{R}^n$ 存在常数 $M > 0$ 使得 $\|\mathbf{prox}_{\alpha h}(x - \alpha y)\| \leqslant M$, 算法 3.8 求解复合优化问题 (3.44) 时, 取 $\gamma_k = \frac{2}{k+1}$, $\beta_k = \frac{1}{2L_\phi}$, $\lambda_k = \frac{k\beta_k}{2}$, 则对任意的 $N \geqslant 1$, 有

$$\min_{k=1,\ldots,N} \|G_{\beta_k}(z_k, \nabla\phi(z_k))\|^2 \leqslant 24L_\phi \left[\frac{4L_\phi\|x_0 - x_*\|^2}{N^2(N+1)} + \frac{L_f}{N}(\|x_*\|^2 + 2M^2)\right]. \tag{3.57}$$

另外, 当 $f(x) = 0$ 时,

$$\Phi(y_N) - \Phi(x_*) \leqslant \frac{4L_g\|x_0 - x_*\|^2}{N(N+1)}. \tag{3.58}$$

由定理 3.8 可以看出, 当目标函数 (3.44) 凸时 ($f(x) = 0$), 算法 3.8 的复杂度与 FISTA 相同, 两者都为 $\mathcal{O}(L_g^2/\sqrt{\epsilon})$. 当目标函数 (3.44) 非凸时 ($f(x) \neq 0$), 算法 3.8 也收敛, 且复杂度为 $\mathcal{O}(L_\phi^{\frac{2}{3}}/\epsilon^{\frac{1}{3}} + L_\phi L_f/\epsilon)$.

4 条件梯度算法的复杂度分析

4.1 引言

考虑带线性约束的优化问题:

$$\min_{x \in X} f(x), \tag{4.1}$$

其中 X 是线性约束集合. 我们可以用投影梯度法求解上述问题. 设上述问题的线性约束集合为 X, 在第 k 步迭代点 x_k 处, 通过极小化目标函数的局部二次逼近来更新

$$x_{k+1} = \operatorname*{argmin}_{x \in X} \left\{ f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2\alpha_k}\|y - x_k\|^2 \right\}, \tag{4.2}$$

即投影梯度迭代

$$x_{k+1} = \mathcal{P}_X(x_k - \alpha_k \nabla f(x_k)). \tag{4.3}$$

注意到约束集合 X 都是线性约束, X 的几何形状是一个多面体. 而子迭代 (4.2) 是一个二次规划问题. 如果线性约束集合 X 很复杂, 则 (4.3) 投影步的计算量很大, 投影梯度法效率会很低. 但是注意到子迭代步 (4.2) 是对优化问题 (4.1) 的一个局部二次逼近, 如果去掉其二次项只做一次逼近, 则子问题 (4.3) 会变成一个线性规划问题:

$$x_{k+1} = \operatorname*{argmin}_{x\in X} \{f(x_k) + \langle \nabla f(x_k), x - x_k\rangle\}. \tag{4.4}$$

这样, 子迭代的复杂度会降低很多. 这就是条件梯度法的核心思想, 通过局部线性逼近来更新 x_{k+1}. 这样不必做约束集合的投影计算, 降低了子问题的计算复杂度.

本节主要用复杂度分析的框架来研究条件梯度法的复杂度. 注意到子迭代 (4.4) 实际上等价于

$$x_{k+1} = \operatorname*{argmin}_{x\in X} \langle \nabla f(x_k), x\rangle. \tag{4.5}$$

在做复杂度分析时, 假设存在一个关于线性约束集合 X 的子程序 $\mathcal{LO}$, 对任意给定向量 $p \in \mathbb{R}^n$ 返回线性规划的解 y, 满足

$$y \in \operatorname*{argmin}_{x\in X} \langle p, x\rangle. \tag{4.6}$$

对任意给定的线性约束集合 X 和向量 $p \in \mathbb{R}^n$, 我们很容易确定子程序 $\mathcal{LO}$ 的计算复杂度 (如内点法求解线性规划的计算复杂度). 因此, 我们可以研究条件梯度法关于子程序 $\mathcal{LO}$ 的分析复杂度 (即条件梯度法为了达到解精度 ϵ 所需调用子程序 $\mathcal{LO}$ 的次数), 以及条件梯度法的复杂度下界, 并且研究加速条件梯度法的可能性.

条件梯度法又被称为 Frank-Wolfe 算法, 其算法原型由 Frank 和 Wolfe [8] 提出, 其发展历史和近几年的研究可以参见文献 [9, 10, 24–31]. 近几年来, 条件梯度法受到了机器学习和优化领域的广泛关注, 主要由于如下原因.

• 子迭代复杂度低: 条件梯度法的子问题是求解一个线性逼近问题, 在很多情形下会比梯度法的非线性逼近子问题要简单. 在下一节里面, 我们会给出带范数不等式约束优化问题的例子, 比较投影梯度法和条件梯度法在求解它们时的效率.

• 简单性: 条件梯度法在实现时不需要精心挑选步长, 且步长不依赖于导数的 Lipschitz 常数. 而梯度法的实现却非常依赖步长的选择, 需要反复地调整以达到收敛的目的. 文献 [29–31] 给出了更详细的解释.

• 解具有结构性: 下一节可以看到, 条件梯度法的解是约束集合 X 极点的凸组合, 因此具有稀疏性和低秩性.

本节主要介绍条件梯度法在凸函数和强凸函数下的复杂度分析, 具有 $\mathcal{LO}$ 子程序的算法在求解约束光滑优化问题时的复杂度下界, 以及利用 Nesterov 加速梯度法框架导出加速条件梯度法, 并分析其复杂度.

4.2 条件梯度法

4.2.1 凸优化问题

本小节研究条件梯度法求解凸函数和强凸函数问题时的收敛性和复杂度, 详细内容可参见文献 [9, 10]. 首先给出条件梯度法求解凸函数优化问题的具体算法 (算法 4.1).

算法 4.1 条件梯度法 (CndG, 凸函数版本)

输入: 给定 $x_0 \in X$, 令 $y_0 = x_0$, 取步长 $\alpha_k \in [0,1]$.
对 $k = 1, \ldots, N$ 执行迭代,
1. 在 y_{k-1} 处调用子程序 $\mathcal{FO}$, 返回 $\nabla f(y_{k-1})$;
2. 在 $\nabla f(y_{k-1})$ 处调用子程序 $\mathcal{LO}$, 返回

$$x_k \in \operatorname*{argmin}_{x \in X} \langle \nabla f(y_{k-1}), x \rangle;$$

3. 令 $y_k = (1-\alpha_k) y_{k-1} + \alpha_k x_k$.
输出: y_N.

条件梯度法有两个迭代点列 $\{x_k\}$ 和 $\{y_k\}$, 其中 $\{y_k\}$ 是收敛点列. 经过简单变化可以得到 $y_k = \operatorname{conv}\{x_0, x_1, \ldots, x_k\}$, 即收敛点 y_k 是点集合 $\{x_0, x_1, \ldots, x_k\}$ 的凸组合. 另外, 由 $y_k = y_{k-1} + \alpha_k(x_k - y_{k-1})$, 条件梯度法可以看作 y_k 沿 $x_k - y_{k-1}$ 的方向按照步长 α_k 进行更新. 为了保证条件梯度法的收敛性, 我们有两种步长的选择方式, 一种是选取严格递减到零的变步长序列:

$$\alpha_k = \frac{2}{k+1}, \quad k = 1, 2, \ldots; \tag{4.7}$$

另一种是采取精确搜索的步长, 求解一个一维的优化问题

$$\alpha_k = \operatorname*{argmin}_{\alpha \in [0,1]} f((1-\alpha) y_{k-1} + \alpha x_k). \tag{4.8}$$

在研究条件梯度法的收敛性和复杂度之前, 我们给出几个条件梯度法比投影梯度法效率高的例子.

例 4.1 (范数约束问题) 考虑带某一范数 $\|\cdot\|$ 约束的凸优化问题

$$\begin{aligned} &\min_x f(x) \\ &\text{s.t. } \|x\| \leqslant t. \end{aligned} \tag{4.9}$$

用条件梯度法求解该问题时, 需要计算子问题

$$x_k \in \operatorname*{argmin}_{\|x\| \leqslant t} \langle \nabla f(y_{k-1}), x \rangle = -t \cdot \Big(\operatorname*{argmax}_{\|x\| \leqslant 1} \langle \nabla f(y_{k-1}), x \rangle \Big) = -t \cdot \partial \|\nabla f(y_{k-1})\|_*, \tag{4.10}$$

其中 $\|z\|_* = \sup\{z^{\mathrm{T}} x, \|x\| \leqslant 1\}$ 是 $\|\cdot\|$ 的对偶范数. 注意到 (4.10) 条件梯度法的子问题相当于计算一个对偶范数的次梯度. 如果计算 $\|\cdot\|$ 范数的次梯度比计算在约束集合 $X = \{x \in \mathbb{R}^n : \|x\| \leqslant t\}$ 上的投影要简单, 那么条件梯度法比投影梯度法效率更高.

例 4.2 (ℓ_1 范数约束问题) 考虑带 ℓ_1 范数约束的凸优化问题

$$\begin{aligned} &\min_x f(x) \\ &\text{s.t. } \|x\|_1 \leqslant t. \end{aligned} \tag{4.11}$$

由于 ℓ_1 范数的对偶范数是 ℓ_∞ 范数, 因此用条件梯度法求解该问题时, 子问题为

$$x_k \in -t \cdot \partial \|\nabla f(y_{k-1})\|_\infty. \tag{4.12}$$

考虑到 ℓ_∞ 范数的次梯度为 $\partial \|x\|_\infty = \{v : \langle v, x \rangle = \|x\|_\infty, \|v\|_1 \leqslant 1\}$, 子问题等价于

$$i_k \in \operatorname*{argmax}_{i=1,\ldots,n} |\nabla_i f(y_{k-1})|, \tag{4.13}$$

$$x_k = -t \cdot \mathbf{sign}[\nabla_{i_k} f(y_{k-1})] \cdot e_{i_k}, \tag{4.14}$$

其中 $\nabla_i f(y_{k-1})$ 表示向量 $\nabla f(y_{k-1})$ 的第 i 个元素, e_i 表示第 i 个元素为 1 的单位向量. 可以看到计算 $\|\cdot\|_\infty$ 的次梯度和计算集合 $X := \{x \in \mathbb{R}^n : \|x\|_1 \leqslant t\}$ 上的投影都需要 $\mathcal{O}(n)$ 的计算复杂度, 但是条件梯度法子问题计算 (4.13) 和 (4.14) 明显要更简单直接.

例 4.3 (ℓ_p 范数约束问题) 考虑带 ℓ_p 范数约束的凸优化问题, 其中 $1 \leqslant p \leqslant \infty$,

$$\begin{aligned} &\min_x f(x) \\ &\text{s.t. } \|x\|_p \leqslant t. \end{aligned} \tag{4.15}$$

由于 ℓ_p 范数的对偶范数是 ℓ_q 范数, 其中 $1/p + 1/q = 1$, 因此用条件梯度法求解该问题时, 子问题为

$$x_k \in -t \cdot \partial \|\nabla f(y_{k-1})\|_q. \tag{4.16}$$

注意到 ℓ_q 范数的次梯度为 $\partial \|x\|_q = \{v : \langle v, x\rangle = \|x\|_q, \|v\|_p \leqslant 1\}$, 子问题等价于

$$x_k^{(i)} = -\beta \cdot \mathbf{sign}[\nabla_i f(y_{k-1})] \cdot |\nabla_i f(y_{k-1})|^{p/q}, \tag{4.17}$$

其中 β 是使得 $\|x_k\|_q = t$ 的归一化常数. 可以看到, 除 $p = 1, 2, \infty$ 这些特殊情形, 条件梯度法的子问题计算复杂度比直接计算点在集合 $X = \{x \in \mathbb{R}^n : \|x\|_p \leqslant t\}$ 上的投影要简单, 后者投影计算需要单独解一个优化问题.

例 4.4 (核范数约束问题) 考虑带核范数约束的矩阵优化问题

$$\begin{aligned} &\min_X f(X) \\ &\text{s.t. } \|X\|_* \leqslant t, \end{aligned} \tag{4.18}$$

其中 $X \in \mathbb{R}^{m\times n}$, 矩阵核范数 $\|\cdot\|_*$ 的对偶范数是其谱范数 $\|\cdot\|_2$, 它们的定义如下:

$$\|X\|_* = \sum_{i=1}^{\min\{m,n\}} \sigma_i(X), \quad \|X\|_2 = \max_{i=1,\ldots,\min\{m,n\}} \sigma_i(X). \tag{4.19}$$

因此, 条件梯度法的子问题为

$$X_k \in -t \cdot \partial \|\nabla f(Y_{k-1})\|_2. \tag{4.20}$$

对矩阵范数的次梯度 $\partial \|X\| = \{Y : \langle Y, X\rangle = \|X\|, \|Y\|_* \leqslant 1\}$, 设 u 和 v 分别是矩阵 $\nabla f(Y_{k-1})$ 最大奇异值对应的左、右奇异向量, 注意到,

$$\langle uv^{\mathrm{T}}, \nabla f(Y_{k-1})\rangle = u^{\mathrm{T}} \nabla f(Y_{k-1}) v = \sigma_{\max}(\nabla f(Y_{k-1})) = \|\nabla f(Y_{k-1})\|_2, \tag{4.21}$$

且 $\|uv^{\mathrm{T}}\|_* = 1$, 因此, 矩阵 $uv^{\mathrm{T}} \in \partial \|\nabla f(Y_{k-1})\|_2$, 则条件梯度法子问题等价于

$$X_k \in -t \cdot uv^{\mathrm{T}}. \tag{4.22}$$

可以看到, 条件梯度法计算子问题时只需要计算矩阵最大的奇异值对应的左、右奇异向量. 如果采用投影梯度法, 其子问题是计算 X 到集合 $\{X \in \mathbb{R}^{m\times n} : \|X\|_* \leqslant t\}$ 的投影, 需要对矩阵做全奇异值分解, 计算量比条件梯度法复杂很多.

现在给出条件梯度法的收敛性和复杂度分析, 首先按照复杂度分析的框架给出问题模型 $\mathcal{F}$. 考虑凸函数和强凸函数两种情形, 对于强凸函数, 条件梯度法无法在理论上建立线性收敛的性质, 因此需要对条件梯度法进行改进, 引入改进的 $\mathcal{LO}$ 子程序. 首先, 给出凸函数情形下的问题模型.

问题模型 4.1 考虑约束凸优化问题集合 $\mathcal{F} = (\Sigma, \mathcal{O}, \mathcal{T}_\epsilon)$:

- 全局信息 Σ: 目标函数 $f \in C_L^{1,1}(X)$, 且 $f(x)$ 是凸函数 (或强凸函数), 约束集合 X 凸且闭有界;
- 局部信息 $\mathcal{O}$: $\mathcal{FO}$ 子程序和 $\mathcal{LO}$ 子程序 (或改进的 $\mathcal{LO}$ 子程序);
- 解的精度 $\mathcal{T}_\epsilon$: 求全局极小点 $\bar{x} \in \mathbb{R}^n$, 使得 $f(\bar{x}) - f^* \leqslant \epsilon$.

衡量条件梯度法的分析复杂度时, 我们需要分别衡量 $\mathcal{FO}$ 和 $\mathcal{LO}$ 子程序的调用次数. 对于凸优化问题, 我们有如下收敛性质和分析复杂度.

定理 4.1 设 $f(x)$ 是凸函数, 条件梯度法 (算法 4.1) 在求解问题模型 4.1 时取步长 (4.7) 或 (4.8), 则产生的序列 $\{x_k\}$ 对任意 $k = 1, 2, \ldots$ 满足

$$f(y_k) - f^* \leqslant \frac{2L}{k(k+1)} \sum_{i=1}^{k} \|x_i - y_{i-1}\|^2 \leqslant \frac{2L}{k+1} D_X^2, \tag{4.23}$$

且为达到 ϵ 精度需要调用 $\mathcal{FO}$ 和 $\mathcal{LO}$ 子程序的次数至多为

$$\left\lceil \frac{2LD_X^2}{\epsilon} \right\rceil - 1. \tag{4.24}$$

证明 令 $\gamma_k = \frac{2}{k+1}$, 记 $\bar{y}_k = (1-\gamma_k)y_{k-1} + \gamma_k x_k$, 则不管 α_k 按下列哪种方式选取:

$$\alpha_k = \frac{2}{k+1} \quad \text{或} \quad \alpha_k = \operatorname*{argmin}_{\alpha \in [0,1]} f((1-\alpha)y_{k-1} + \alpha x_k),$$

对 $y_k = (1-\alpha_k)y_{k-1} + \alpha_k x_k$, 都有 $f(y_k) \leqslant f(\bar{y}_k)$. 注意到 $\bar{y}_k - y_{k-1} = \gamma_k(x_k - y_{k-1})$, 由 $f(x) \in C_L^{1,1}(X)$, 利用性质 2.6, 有

$$\begin{aligned} f(y_k) &\leqslant f(\bar{y}_k) \leqslant f(y_{k-1}) + \langle \nabla f(y_{k-1}), \bar{y}_k - y_{k-1} \rangle + \frac{L}{2}\|\bar{y}_k - y_{k-1}\|^2 \\ &\leqslant (1-\gamma_k) f(y_{k-1}) + \gamma_k [f(y_{k-1}) + \langle \nabla f(y_{k-1}), x - y_{k-1} \rangle] + \frac{L\gamma_k^2}{2}\|x_k - y_{k-1}\|^2 \\ &\leqslant (1-\gamma_k) f(y_{k-1}) + \gamma_k f(x) + \frac{L\gamma_k^2}{2}\|x_k - y_{k-1}\|^2, \quad \text{对任意 } x \in X, \end{aligned} \tag{4.25}$$

其中不等式 (4.25) 的第 3 个小于等于号成立是因为 $x_k \in \min_{x \in X} \langle \nabla f(y_{k-1}), x \rangle$, 由最优性条件可以得到对任意 $x \in X$, 有 $\langle x - x_k, \nabla f(y_{k-1}) \rangle \geqslant 0$. 将不等式 (4.25) 稍做变换, 对任意 $x \in X$, 有

$$f(y_k) - f(x) \leqslant (1-\gamma_k)[f(y_{k-1}) - f(x)] + \frac{L}{2}\gamma_k^2 \|x_k - y_{k-1}\|^2. \tag{4.26}$$

由引理 2.1 有

$$f(y_k) - f(x) \leqslant \Gamma_k (1-\gamma_1)[f(y_0) - f(x)] + \frac{\Gamma_k L}{2} \sum_{i=1}^{k} \frac{\gamma_i^2}{\Gamma_i} \|x_i - y_{i-1}\|^2. \tag{4.27}$$

由 $\gamma_k = \frac{2}{k+1}$ 和 $\gamma_1 = 1$ 得到 $\Gamma_k = \frac{2}{k(k+1)}$, 从而可以得到收敛性不等式

$$f(y_k) - f^* \leqslant \frac{2L}{k(k+1)} \sum_{i=1}^{k} \|x_i - y_{i-1}\|^2 \leqslant \frac{2L}{k+1} D_X^2. \tag{4.28}$$

令 $\frac{2L}{k+1} D_X^2 \leqslant \epsilon$, 可以得到分析复杂度结论. □

通过定理 4.1 可以看到, 首先, 条件梯度法寻找 ϵ 近似解所需要调用子程序 $\mathcal{LO}$ 和 $\mathcal{FO}$ 的次数相同, 都不超过

$$\mathcal{O}\left(\frac{LD_X^2}{\epsilon}\right). \tag{4.29}$$

在下一节可以看到, 在函数集合 $C_L^{1,1}(X)$ 条件下, 条件梯度法关于子程序 $\mathcal{LO}$ 的调用次数 $\mathcal{O}(1/\epsilon)$ 达到了算法复杂度下界, 因此是最优的. 但 $\mathcal{FO}$ 的调用次数 $\mathcal{O}(1/\epsilon)$ 并没有达到 $C_L^{1,1}(X)$ 函数集合的复杂度下界 $\mathcal{O}(1/\sqrt{\epsilon})$. 其次, 注意到 Lipschitz 常数 L 和 D_X 都与设定的范数 $\|\cdot\|$ 有关, 即 $L \equiv L_{\|\cdot\|}$, $D_X \equiv D_{X,\|\cdot\|}$. 尽管条件梯度法的分析复杂度结论 (4.29) 与范数无关, 但实际上, 可以利用这一性质选择合适的范数, 使得分析复杂度关于范数达到极小

$$\mathcal{O}(1)\inf_{\|\cdot\|}\left\{\frac{L_{\|\cdot\|}D_{X,\|\cdot\|}}{\epsilon}\right\}. \tag{4.30}$$

例如, 当约束集合 X 是单纯形时, 可以选取 $\|\cdot\| = \|\cdot\|_1$. 最后, 注意到 (4.23) 中收敛速度依赖于 $\|x_i - y_{i-1}\|^2$, 但实际上其值并不一定随着 k 增加而趋于 0. 例如, 设 X 是多面体, 则由子问题是线性规划, 知道其解 x_i 是 X 的顶点. 考虑到 $\{y_i\} \to y^*$, 其中 x^* 是最优解, 除非 x^* 也是 X 多面体的顶点, 则当 k 足够大时, $\|x_i - y_{i-1}\|$ 并不会趋于 0.

4.2.2 强凸优化问题

本小节考虑条件梯度法求解强凸优化问题的收敛性. 为此, 我们需要对 $\mathcal{LO}$ 子程序做一些改进, 从而改善条件梯度法在求解强凸问题时的 $\mathcal{LO}$ 分析复杂度. 更具体地, 对改进的 $\mathcal{LO}$ 子程序输入某一给定的 $p \in \mathbb{R}^n$ 和 $x_0 \in X$, 能够返回下列优化问题的最优解:

$$\min\{\langle p, x\rangle :\ x \in X, \|x - x_0\| \leqslant R\}. \tag{4.31}$$

改进的 $\mathcal{LO}$ 子程序的复杂度与范数的选择有关. 实际上, 可以选择特定的范数来降低优化问题 (4.31) 的复杂度. 例如, 当 X 是多面体时, 我们可以取 $\|\cdot\| = \|\cdot\|_\infty$ 或 $\|\cdot\| = \|\cdot\|_1$, 则改进的 $\mathcal{LO}$ 子程序与原 $\mathcal{LO}$ 的复杂度相同, 都是求解一个线性规划问题. 但是 $\|\cdot\|$ 的选取会改变 L 和 μ 的值. 在改进的 $\mathcal{LO}$ 基础上, 文献 [10] 给出了求解强凸优化问题的收缩条件梯度法 (shrinking conditional gradient (S-CndG) method), 见算法 4.2.

算法 4.2 收缩条件梯度法 (S-CndG)

输入: $p_0 \in X$, 令 $R_0 = D_X$.
对 $t = 1, 2, \ldots$ 执行迭代,
令 $y_0 = p_{t-1}$.
对 $k = 1, \ldots, 8L/\mu$ 执行迭代,
1. 调用改进的 $\mathcal{LO}$ 子程序计算

$$x_k \in \operatorname*{argmin}_{x \in X_{t-1}} \langle \nabla f(y_{k-1}), x\rangle,$$

其中 $X_{t-1} := \{x \in X : \|x - p_{t-1}\| \leqslant R_{t-1}\}$;
2. 令 $y_k = (1 - \alpha_k)y_{k-1} + \alpha_k x_k$, 其中 $\alpha_k \in [0, 1]$.
令 $p_t = y_k$ 和 $R_t = R_{t-1}/\sqrt{2}$.

定理 4.2 设 $f(x) \in \mathcal{F}_{L,\mu}^{1,1}$, 收缩条件梯度法 4.2 在求解问题模型 4.1 时取步长 (4.7) 或 (4.8), 产

生的序列 $\{p_t\}$ 对任意 $t = 1, 2, \ldots$ 满足

$$f(p_t) - f^* \leqslant \frac{\mu R_0}{2^t}, \tag{4.32}$$

且为达到 ϵ 精度需要调用 $\mathcal{FO}$ 和改进的 $\mathcal{LO}$ 子程序的次数至多是

$$\frac{8L}{\mu}\left\lceil \max\left(\log_2 \frac{\mu R_0}{\epsilon}, 1\right)\right\rceil. \tag{4.33}$$

证明 令 $K \equiv 8L/\mu$, 首先证明对任意的 $t \geqslant 0$ 有 $x^* \in X_t$. 使用归纳法, 当 $t = 0$ 时,

$$\|x^* - p_0\| \leqslant R_0 = D_X,$$

因此, $x^* \in X_0$. 假设对任意 $t \geqslant 1$, $x^* \in X_{t-1}$ 成立. 在定理 4.1 的证明过程中令 $X = X_t$, 可知第 t 次外迭代, 对 $k = 1, \ldots, K$, 有

$$f(y_k) - f^* \leqslant \frac{2L}{k(k+1)} \sum_{i=1}^{k} \|x_i - y_{i-1}\|^2 \leqslant \frac{2L}{k+1} R_{t-1}^2, \quad k = 1, \ldots, K. \tag{4.34}$$

当 $k = K$ 时, $p_t = y_K$, 由强凸函数的性质 $f(y_K) - f^* \geqslant \frac{\mu}{2}\|y_K - x^*\|^2$, 有

$$\|p_t - x^*\|^2 \leqslant \frac{2}{\mu}[f(p_t) - f^*] \leqslant \frac{4L}{\mu(K+1)} R_{t-1}^2 \leqslant \frac{1}{2} R_{t-1}^2 = R_t^2. \tag{4.35}$$

因此, 我们证明了 $x^* \in X_t$ 对任意 $t \geqslant 0$ 成立. 我们现在可以给出 ϵ 近似解所需要的 $\mathcal{LO}$ 和 $\mathcal{FO}$ 上界. 注意到 (4.35) 以及 R_t 的定义, 有

$$f(p_t) - f^* \leqslant \frac{\mu}{2}\, R_t^2 = \frac{\mu R_0}{2^t}. \tag{4.36}$$

因此为得到 ϵ 近似解所需要的外迭代次数不超过 $\lceil \max\{\log_2(\mu R_0/\epsilon), 1\}\rceil$, 考虑到内迭代次数恒定为 K, 可以证明改进的 $\mathcal{LO}$ 子程序调用次数不超过 (4.33). □

4.2.3 加速框架下的条件梯度法

本小节在加速梯度法框架中引入条件梯度法的 $\mathcal{LO}$ 子程序, 具体地, 在加速梯度法中, 将非线性逼近子问题修改为线性逼近的子问题, 从而可以得到不同的收敛性结果. 加速框架下的条件梯度法 (primal averaging conditional gradient (PA-CndG) method) 由文献 [10] 提出, 具体算法见算法 4.3.

算法 4.3 加速框架下的条件梯度法 (PA-CndG)

输入: $x_0 \in X$, 步长序列 $\{\alpha_k\}$, $\alpha_k \in [0, 1]$, 令 $y_0 = x_0$.
对 $k = 1, 2, \ldots$ 执行迭代,
1. $z_k = \frac{k-1}{k+1} y_{k-1} + \frac{2}{k+1} x_{k-1}$;
2. 令 $p_k = \nabla f(z_k)$, 调用 $\mathcal{LO}$ 子程序计算

$$x_k \in \operatorname*{argmin}_{x \in X} \langle p_k, x\rangle;$$

3. $y_k = (1 - \alpha_k) y_{k-1} + \alpha_k x_k$.
输出

定理 4.3 算法 4.3 在求解问题模型 4.1 时取步长 (4.7) 或 (4.8), 则产生的序列 $\{x_k\}$ 和 $\{y_k\}$ 对任意 $k=1,2,\ldots$ 满足

$$f(y_k)-f^*\leqslant\frac{2L}{k(k+1)}\sum_{i=1}^{k}\|x_i-x_{i-1}\|^2\leqslant\frac{2L}{k+1}D_X^2. \tag{4.37}$$

证明 令 $\gamma_k=\frac{2}{k+1}$, 记 $\bar{y}_k=(1-\gamma_k)y_{k-1}+\gamma_k x_k$, 则不管 α_k 按 (4.7) 或 (4.8) 哪种方式选取, 对 $y_k=(1-\alpha_k)y_{k-1}+\alpha_k x_k$, 都有 $f(y_k)\leqslant f(\bar{y}_k)$. 由算法 4.3 的定义, 有 $z_{k-1}=(1-\gamma_k)y_{k-1}+\gamma_k x_{k-1}$, 因此, $\bar{y}_k-z_{k-1}=\gamma_k(x_k-x_{k-1})$. 由性质 2.6 有

$$\begin{aligned}
f(y_k)&\leqslant f(\bar{y}_k)\leqslant f(z_{k-1})+\langle\nabla f(z_{k-1}),\bar{y}_k-z_{k-1}\rangle+\frac{L}{2}\|\bar{y}_k-z_{k-1}\|^2\\
&=(1-\gamma_k)[f(z_{k-1})+\langle\nabla f(z_{k-1}),y_{k-1}-z_{k-1}\rangle]\\
&\quad+\gamma_k[f(z_{k-1})+\langle\nabla f(z_{k-1}),x_k-z_{k-1}\rangle]+\frac{L\gamma_k^2}{2}\|x_k-x_{k-1}\|^2\\
&\leqslant(1-\gamma_k)f(y_{k-1})+\gamma_k[f(z_{k-1})+\langle\nabla f(z_{k-1}),x-z_{k-1}\rangle]+\frac{L\gamma_k^2}{2}\|x_k-x_{k-1}\|^2\\
&\leqslant(1-\gamma_k)f(y_{k-1})+\gamma_k f(x)+\frac{L\gamma_k^2}{2}\|x_k-x_{k-1}\|^2,\quad\text{对任意 } x\in X,
\end{aligned}\tag{4.38}$$

其中不等式 (4.38) 的第 3 个小于等于号成立是利用了 f 的凸性以及 $x_k\in\min_{x\in X}\langle\nabla f(z_{k-1}),x\rangle$ 的最优性条件. 由最优性条件, 对任意的 $x\in X$, 不等式 $\langle x-x_k,\nabla f(z_{k-1})\rangle\geqslant 0$ 成立. 将不等式 (4.38) 两边同时减去 $f(x)$, 对任意的 $x\in X$, 有

$$f(y_k)-f(x)\leqslant(1-\gamma_k)[f(y_{k-1})-f(x)]+\frac{L}{2}\gamma_k^2\|x_k-x_{k-1}\|^2. \tag{4.39}$$

由引理 2.1, 有

$$f(y_k)-f(x)\leqslant\Gamma_k(1-\gamma_1)[f(y_0)-f(x)]+\frac{\Gamma_k L}{2}\sum_{i=1}^{k}\frac{\gamma_i^2}{\Gamma_i}\|x_i-x_{i-1}\|^2. \tag{4.40}$$

由 $\gamma_k=\frac{2}{k+1}$ 和 $\gamma_1=1$ 得到 $\Gamma_k=\frac{2}{k(k+1)}$, 于是, 对任意 $x\in X$, 有

$$f(y_k)-f^*\leqslant\frac{2L}{k(k+1)}\sum_{i=1}^{k}\|x_i-x_{i-1}\|^2\leqslant\frac{2L}{k+1}D_X^2. \tag{4.41}$$

上述收敛性不等式成立. □

比较条件梯度法的收敛性结果 (4.23) 和加速框架下的条件梯度法收敛性结果 (4.37), 我们可以发现: 对于条件梯度法, 收敛性速度与 $\|x_k-y_{k-1}\|$ 的变化有关, 而 $\|x_k-y_{k-1}\|$ 不一定随着 k 增加收敛到 0; 对于加速框架下的条件梯度法, 收敛速度与 $\|x_k-x_{k-1}\|$ 有关, 而 $\|x_k-x_{k-1}\|$ 是否随 k 增加收敛到 0 依赖于约束集合 X 的结构以及 $\|p_{k+1}-p_k\|$. 令 $\gamma_k=\frac{2}{k+1}$, 注意到 $z_k=(1-\gamma_k)y_{k-1}+\gamma_k x_{k-1}$, $y_k=(1-\alpha_k)y_{k-1}+\alpha_k x_k$, 因此,

$$\begin{aligned}
z_{k+1}-z_k&=(y_k-y_{k-1})+\gamma_{k+1}(x_k-y_{k-1})+\gamma_k(x_{k-1}-y_{k-1})\\
&=\alpha_k(x_k-y_{k-1})+\gamma_{k+1}(x_k-y_{k-1})+\gamma_k(x_{k-1}-y_{k-1}).
\end{aligned}\tag{4.42}$$

令 $\alpha_k = \gamma_k$, 则有 $\|z_{k+1} - z_k\| \leqslant 3\gamma_k D_X$, 注意到 $f(x) \in C_L^{1,1}(X)$,

$$\|p_{k+1} - p_k\|_* = \|\nabla f(z_{k+1}) - \nabla f(z_k)\|_* \leqslant 3\gamma_k L D_X. \tag{4.43}$$

因此, p_{k+1} 和 p_k 的差随着 k 增加趋向 0 (因为 γ_k 随着 k 增加单调递减到 0). 我们可以加入一些关于 $\mathcal{LO}$ 更强的假设 (在某些局部下满足) 来提高 PA-CndG 的收敛性. 例如,

$$\|x_k - x_{k-1}\| \leqslant Q\|p_k - p_{k-1}\|_*^\rho, \quad \text{对某一 } k \geqslant N. \tag{4.44}$$

文献 [10] 给出了类似 (4.44) 条件下加速框架下条件梯度法的收敛速度. 可以看到, 在此类条件下, 收敛速度有了明显的提高.

定理 4.4 算法 4.3 取步长 $\alpha_k = \frac{2}{k+1}$, 假设子程序 $\mathcal{LO}$ 满足

$$\|x_k - x_{k-1}\| \leqslant Q\|p_k - p_{k-1}\|_*^\rho, \quad k \geqslant 2. \tag{4.45}$$

对某些给定的 $\rho \in (0, 1]$ 和 $Q > 0$, 对任意的 $k \geqslant 1$, 有如下收敛性结果:

$$f(y_k) - f^* \leqslant \mathcal{O}(1) \begin{cases} \dfrac{Q^2 L^{2\rho+1} D_X^{2\rho}}{(1-2\rho)k^{2\rho+1}}, & \rho \in (0, 0.5), \\ \dfrac{Q^2 L^2 D_X \log(k+1)}{k^2}, & \rho = 0.5, \\ \dfrac{Q^2 L^{2\rho+1} D_X^{2\rho}}{(2\rho-1)k^2}, & \rho \in (0.5, 1]. \end{cases}$$

4.3 复杂度下界

本小节研究问题模型 4.1 关于 $\mathcal{LO}$ 子程序的分析复杂度下界. 首先定义带 $\mathcal{LO}$ 子程序的解算法 $\mathcal{S}$, 考虑其最一般的框架形式 (算法 4.4).

算法 4.4 条件梯度法的一般框架

输入: $x_0 \in X$.
对 $k = 1, \ldots, N$ 执行迭代,
1. 确定线性逼近函数 $\langle p_k, \cdot \rangle$;
2. 调用 $\mathcal{LO}$ 子程序计算 $x_k \in \operatorname{argmin}_{x \in X} \langle p_k, x \rangle$;
3. 输出 $y_k \in \operatorname{conv}\{x_0, \ldots, x_k\}$.

注意到上述算法框架的变化形式很多. 首先, 算法 4.1 和 4.3 都是其特殊情形. 其次, 对于线性逼近函数 $\langle p_k, \cdot \rangle$ 没有任意限制. 例如, 当 f 光滑时, p_k 可以是某个可行点处的梯度, 或者是历史梯度的一些线性组合. 当 f 非光滑时, p_k 可以是 f 光滑逼近函数的梯度. 我们也可以考虑 p_k 包含噪声, 或者一些二阶梯度信息. 最后, 输出的解 y_k 是点集合 $\{x_0, \ldots, x_k\}$ 的凸组合, 因此, 输出的解可以有多种多样的组合方式, 与 $\{x_k\}$ 中任意一个点都不同.

将上述算法框架与一阶算法进行比较可以发现, 一方面, 条件梯度法的一般框架求解线性逼近的子问题, 而一阶算法需要求解非线性子问题 (投影或近似点算子的计算); 另一方面, 条件梯度法的一般框架对于算法的迭代方向 p_k 和输出的点 y_k 比一阶算法更为灵活.

为了研究优化问题关于 $\mathcal{LO}$ 子程序分析复杂度下界, 按照复杂度分析理论, 我们需要定义问题模型 $\mathcal{F}$、解算法 $\mathcal{S}$ 和解算法集合 $\mathcal{M}$. 对于问题模型 $\mathcal{F}$, 考虑其中全局信息为 $f \in C_{L,\|\cdot\|}^{1,1}(X)$ 且是凸函数

的情形. 具体的解算法 $\mathcal{S}$ 定义为算法 4.4 框架里可以将问题模型 $\mathcal{F}$ 求解到 ϵ 精度的算法. 解算法集合 $\mathcal{M}$ 定义为所有这样形式的解算法 $\mathcal{S}$ 组成的集合. 因此需要估计

$$\mathrm{Compl}(\epsilon) = \inf_{\mathcal{S}\in\mathcal{M}} \mathrm{Compl}_{\mathcal{S}}(\epsilon), \tag{4.46}$$

其中 $\mathrm{Compl}_{\mathcal{S}}(\epsilon)$ 是解算法 $\mathcal{S}$ 求解问题集合 $\mathcal{F}$ 时 $\mathcal{LO}$ 分析复杂度的上界. 我们无法直接计算出 $\mathrm{Compl}(\epsilon)$, 只能给出 $\mathrm{Compl}(\epsilon)$ 的一个下界估计. 但按照复杂度分析理论 [3, 6], 我们需要构造 $\mathcal{F}$ 中的一个函数, 对该函数调用子程序 $\mathcal{LO}$ 所获得的局部信息很少, 导致解算法集合 $\mathcal{M}$ 中的所有算法在求解它时的效率都不高. 文献 [10] 给出了如下复杂度下界结果.

定理 4.5 (光滑优化问题的 $\mathcal{LO}$ 复杂度下界) 为了达到 $\epsilon > 0$ 近似解, 解算法集合 $\mathcal{M}$ 中的任意解算法 $\mathcal{S}$ 在求解全局信息为凸函数和 $C^{1,1}_{L,\|\cdot\|}(X)$ 的问题模型 $\mathcal{F}$ 时所调用的 $\mathcal{LO}$ 子程序次数 $\mathrm{Compl}_{\mathcal{S}}(\epsilon)$ 满足

$$\mathrm{Compl}_{\mathcal{S}}(\epsilon) \geqslant \left\lfloor \min\left\{\frac{n}{4}, \frac{LD_X^2}{8\epsilon}\right\}\right\rfloor - 1 \tag{4.47}$$

对任意 $\mathcal{S}\in\mathcal{M}$ 成立.

证明 考虑凸优化问题

$$f^* = \min_{x\in X}\left\{f(x) := \frac{L}{2}\sum_{i=1}^{n}(x^{(i)})^2\right\}, \tag{4.48}$$

其中 $x^{(i)}$ 是 x 的第 i 个元素, $X := \{x\in\mathbb{R}^n : \sum_{i=1}^n x^{(i)} = D, x^{(i)} > 0\}$, $D > 0$, 容易看到 (4.48) 的最优解 x^* 和最优值 f^* 分别为

$$x^* = \left\{\frac{D}{n}, \ldots, \frac{D}{n}\right\}, \quad f^* = \frac{LD^2}{n}, \tag{4.49}$$

且函数 $f(x)\in C^{1,1}_{L,\|\cdot\|}(X)$, 其中 $\|\cdot\|$ 取 $\|\cdot\|_2$. 不失一般性, 我们可以假设算法的初始点是 $x_0 = De_1$, 其中 $e_1 = (1, 0, \ldots, 0)$ 是单位向量. 如果选择随机初始点 $x_0\in X$, 我们可以构造相似问题

$$\begin{aligned} &\min_x (x^{(1)})^2 + \sum_{i=2}^{n}(x^{(i)} - x_0^{(i)})^2 \\ &\text{s.t. } x^{(1)} + \sum_{i=2}^{n}(x^{(i)} - x_0^{(i)}), \\ &\qquad x^{(1)} \geqslant 0, \\ &\qquad x^{(i)} - x_0^{(i)} \geqslant 0, \quad i = 2, \ldots, n. \end{aligned} \tag{4.50}$$

然后对该问题进行分析.

现用算法集合 $\mathcal{M}$ 中的解算法 $\mathcal{S}$ 求解 (4.48). 在第 k 次迭代, 算法调用 $\mathcal{LO}$ 子程序, 对任意的输入 p_k 输出新的迭代点 $x_k\in\mathrm{Argmin}_{x\in X}\langle p_k, x\rangle$. 由于 X 是单纯形, 则 $x_k\in\{De_1, De_2, \ldots, De_n\}$, e_i $(i = 1, \ldots, n)$ 是 $\mathbb{R}^n$ 第 i 个单位向量. 注意到 x_k 会有重复相同的情形, 记 $x_k = De_{p_k}$, 其中 $1\leqslant p_k\leqslant n$. 由算法 4.4 定义有 $y_k\in D\ \mathrm{Conv}\{x_0, x_1, \ldots, x_k\}$. 因此,

$$y_k\in D\ \mathrm{Conv}\{e_1, e_{p_1}, \ldots, e_{p_k}\}. \tag{4.51}$$

假设 $\{e_1, e_{p_1}, \ldots, e_{p_k}\}$ 中 q 个向量线性独立, 其中 $1\leqslant q\leqslant k+1\leqslant n$. 不失一般性, 记 $\{e_1, e_{p_1}, \ldots, e_{p_{q-1}}\}$ 线性独立. 因此,

$$f(y_k) \geqslant \min_x\{f(x) : x\in D\cdot\mathrm{conv}\{e_1, e_{p_1}, \ldots, e_{p_k}\}\}$$

$$= \min_x\{f(x) : x \in D \cdot \mathrm{conv}\{e_1, e_{p_1}, \ldots, e_{p_{q-1}}\}\}$$
$$= \frac{LD^2}{2q} \geqslant \frac{LD^2}{2(k+1)}. \tag{4.52}$$

上述不等式与 (4.49) 结合, 对任意 $k = 1, \ldots, n-1$,

$$f(y_k) - f^* \geqslant \frac{LD^2}{2(k+1)} - \frac{LD^2}{n} \tag{4.53}$$

成立. 注意到 $D_X = \sqrt{2D}$, 定义

$$K := \left\lfloor \min\left\{\frac{n}{4}, \frac{LD_X^2}{8\epsilon}\right\}\right\rfloor - 1. \tag{4.54}$$

由 (4.53), 对任意的 $1 \leqslant k \leqslant K$, 有

$$\begin{aligned} f(y_k) - f^* &\geqslant \frac{LD^2}{2(K+1)} - \frac{LD^2}{n} \\ &\geqslant \frac{LD^2}{\min\{\frac{n}{2}, \frac{LD^2}{2\epsilon}\}} - \frac{LD^2}{n} \\ &= \frac{LD^2}{\min\{n, \frac{LD^2}{\epsilon}\}} + \left(\frac{LD^2}{\min\{n, \frac{LD^2}{\epsilon}\}} - \frac{LD^2}{n}\right) \\ &\geqslant \frac{LD^2}{\frac{LD^2}{\epsilon}} + \left(\frac{LD^2}{n} - \frac{LD^2}{n}\right). \end{aligned}$$

证毕. □

由上述定理可以看出, 当 n 足够大时, 光滑优化问题的 $\mathcal{LO}$ 复杂度下界为

$$\mathcal{O}\left(\frac{LD_X^2}{\epsilon}\right). \tag{4.55}$$

4.4 加速条件梯度法

利用加速梯度法框架, 同样可以得到加速的条件梯度法. 这里的加速, 指的是 $\mathcal{FO}$ 的调用次数减少, 算法效率的提高. 从上一节可知条件梯度法已经在 $\mathcal{LO}$ 的调用次数上达到了最优, 但对于 $\mathcal{FO}$ 的调用次数并没有达到最优. 因此, 我们可以利用加速梯度法的框架来减少对 $\mathcal{FO}$ 的调用次数. 文献 [9] 给出了加速条件梯度法的框架, 又称为条件梯度滑动算法 (conditional gradient sliding (CGS) method), 见算法 4.5.

算法 4.5 加速条件梯度法

输入: $x_0 \in X$ 和总迭代次数 N, 取 $\beta_k > 0$, $\eta_k \geqslant 0$, $\gamma_k \in [0, 1]$, 令 $y_0 = x_0$.

对 $k = 1, \ldots, N$ 执行迭代,

$$z_k = (1 - \gamma_k)y_{k-1} + \gamma_k x_{k-1}, \tag{4.56}$$
$$x_k = \mathrm{CndG}(f'(z_k), x_{k-1}, \beta_k, \eta_k), \tag{4.57}$$
$$y_k = (1 - \gamma_k)y_{k-1} + \gamma_k x_k. \tag{4.58}$$

输出: y_N.

加速条件梯度法也有 3 个迭代序列 $\{x_k\}$、$\{y_k\}$ 和 $\{z_k\}$, 以及 3 个待定的参数序列 $\{\gamma_k\}$、$\{\beta_k\}$ 和 $\{\eta_k\}$. 与算法 2.3 不同之处在于算法 4.5 第二步中 x_k 的更新方式. 加速梯度法通过直接精确求解非线性优化问题得到 x_k, 而加速条件梯度法实际上是通过非精确求解非线性优化问题来更新 x_k, 具体来说, 加速条件梯度法是用条件梯度法来求解非线性优化的子问题, 其中更新 x_k 的子程序 CndG 如下:

子程序 CndG $u^+ = \mathrm{CndG}(g, u, \beta, \eta)$

1. 令 $u_1 = u$ 和 $t = 1$;
2. 调用 $\mathcal{LO}$ 求解下面子问题, 令 v_t 表示其最优解,

$$V_{g,u,\beta}(u_t) = \max_{x\in X}\langle g + \beta(u_t - u_1), u_t - x\rangle; \tag{4.59}$$

3. 若 $V_{g,u,\beta}(u_t) \leqslant \eta$, 令 $u^+ = u_t$ 并且停机;
4. 令 $u_{t+1} = (1-\alpha_t)u_t + \alpha_t v_t$, 其中

$$\alpha_t = \min\left\{1, \frac{\langle\beta(u-u_t) - g, v_t - u_t}{\beta\|v_t - u_t\|^2}\right\}; \tag{4.60}$$

5. 取 $t \leftarrow t+1$ 转步 2.

可以看到子程序 CndG 的收敛点列是 $\{u_t\}$. 记 $\phi(x) := \langle g, x\rangle + \frac{\beta}{2}\|x-u\|^2$, 子程序 CndG 可以被看成是利用条件梯度法求解 $\min_{x\in X}\phi(x)$. 注意到 $\nabla\phi(x) = g + \beta\langle x - u\rangle$, 因此, (4.59) 等价于

$$v_t \in \operatorname*{argmax}_{x\in X}\langle\nabla\phi(u_t), u_t - x\rangle = \operatorname*{argmin}_{x\in X}\langle\nabla\phi(u_t), x\rangle. \tag{4.61}$$

因此可以写出子程序 CndG 更新 x_k 的等价形式. 求解优化问题

$$\min_{x\in X}\left\{\phi_(x) := \langle g, x\rangle + \frac{\beta_k}{2}\|x - u_1\|^2\right\}, \tag{4.62}$$

其中 $g = \nabla f(z_k)$, $u_1 = x_{k-1}$, 对 $t = 1, 2, \ldots$ 执行迭代,

(1) $v_t = \operatorname{argmin}_{x\in X}\langle\nabla\phi(u_t), x\rangle$;

(2) $u_{t+1} = (1-\alpha_t)u_t + \alpha_t v_t$;

(3) 当 $V_{g,u_1,\beta}(u_t) := \max_{x\in X}\langle\nabla\phi(u_t), u_t - x\rangle \leqslant \eta$ 时停止迭代, 其中 $V_{g,u,\beta}$ 又被称为 Wolfe 间隔, 表示约束优化问题最优性条件的满足程度. 步长 α_t 的选取实际上等价于下面的优化问题:

$$\alpha_t = \operatorname*{argmin}_{\alpha\in[0,1]}\phi((1-\alpha)u_t + \alpha v_t). \tag{4.63}$$

可以看出, 加速条件梯度法的迭代点 x_k 是非线性投影子问题

$$\min_{x\in X}\left\{\phi_k(x) := \langle\nabla f(z_k), x\rangle + \frac{\beta_k}{2}\|x - x_{k-1}\|^2\right\} \tag{4.64}$$

满足约束条件

$$\langle\nabla\phi_k(x_k), x_k - x\rangle := \langle\nabla f(z_k) + \beta_k\langle x_k - x_{k-1}, x_k - x\rangle \leqslant \eta_k, \quad \forall\, x\in X \tag{4.65}$$

的近似解.

为了证明加速条件梯度法的收敛性和复杂度, 我们采取分步的策略. 第一步, 确定子程序 CndG 的收敛性和复杂度; 第二步, 确定外迭代的收敛性和复杂度. 记 ϕ_k^* 表示第 k 步所近似求解的非线性投影问题 (4.64) 的精确解. 首先考虑内迭代子程序 CndG 的收敛性和复杂度.

子程序 CndG 的复杂度:

首先, 考虑子程序求解的优化问题 (4.64) $\min_{x\in X}\phi_k(x)$ 的收敛性, 即 $\phi_k(u_t)-\phi_k^*$ 的收敛性和复杂度. 为了简化记号, 我们在分析中省略外迭代下标 k, 即用 $\phi(x)$ 代替 $\phi_k(x)$, 用 β 和 η 分别代替 β_k 和 η_k. 利用引理 2.1 的分析方法, 构造序列 $\{\lambda_t\}$ 和 $\{\Gamma_t\}$, 满足

$$\Lambda_{t+1}=\Lambda_t(1-\lambda_{t+1}),\quad \forall\, t\geqslant 2. \tag{4.66}$$

我们可以取 $\lambda_t:=2/t$, 于是 $\Lambda_t=2/[t(t-1)]$. 令 $\bar{u}_{t+1}=(1-\lambda_{t+1})u_t+\lambda_{t+1}v_t$, 可以推出

$$\bar{u}_{t+1}-u_t=\lambda_{t+1}(v_t-u_t).$$

注意到 $u_{t+1}=(1-\alpha_t)u_t+\alpha_t v_t$ 和 $\alpha_t=\operatorname{argmin}_{\alpha\in[0,1]}\phi((1-\alpha)u_t+\alpha v_t)$, 可以推出 $\phi(u_{t+1})\leqslant\phi(\bar{u}_{t+1})$. 由 $\phi(x)\in C_L^{1,1}(X)$, 利用引理 2.1, 有

$$\begin{aligned}\phi(u_{t+1})&\leqslant\phi(\bar{u}_{t+1})\leqslant\phi(u_t)+\langle\phi'(u_t),\bar{u}_{t+1}-u_t\rangle+\frac{\beta}{2}\|\bar{u}_{t+1}-u_t\|^2\\&=\phi(u_t)+\lambda_{t+1}\langle\phi'(u_t),v_t-u_t\rangle+\frac{\beta\lambda_{t+1}^2}{2}\|v_t-u_t\|^2\\&\leqslant(1-\lambda_{t+1})\phi(u_t)+\lambda_{t+1}[\phi(u_t)+\langle\phi'(u_t),v_t-u_t\rangle]+\frac{\beta\lambda_{t+1}^2}{2}\|v_t-u_t\|^2\\&\leqslant(1-\lambda_{t+1})\phi(u_t)+\lambda_{t+1}\phi(x)+\frac{\beta\lambda_{t+1}^2}{2}\|v_t-u_t\|^2.\end{aligned}\tag{4.67}$$

上面不等式两边同时减去 $\phi(x)$ 可以得到 $\phi_k(u_{t+1})-\phi_k^*$ 误差的估计不等式

$$\phi(u_{t+1})-\phi(x)\leqslant(1-\lambda_{t+1})[\phi(u_t)-\phi(x)]+\frac{\beta\lambda_{t+1}^2}{2}\|v_t-u_t\|^2,\quad\forall\, x\in X. \tag{4.68}$$

将上式从 1 到 t 求和得到

$$\phi(u_{t+1})-\phi(x)\leqslant\Lambda_{t+1}(1-\lambda_2)[\phi(u_1)-\phi(x)]+\Lambda_{t+1}\beta\sum_{j=1}^{t}\frac{\lambda_{j+1}^2}{2\Lambda_{j+1}}\|v_j-u_j\|^2\leqslant\frac{2\beta D_X^2}{t+1}. \tag{4.69}$$

取 $x=x^*$, 我们得到 $\phi_k(u_t)-\phi_k^*$ 的收敛速度估计为

$$\phi(u_{t+1})-\phi^*\leqslant\frac{2\beta D_X^2}{t+1}. \tag{4.70}$$

有了收敛速度, 我们可以估计子程序的复杂度. 注意到子程序 CndG 的复杂度的停止条件为 $V_{g,u_1,\beta}(u_t)\leqslant\eta$, 因此需要建立 $V_{g,u_1,\beta}(u_t)$ 与 $\phi(u_t)-\phi^*$ 之间的关系. 记 $\Delta_t:=\phi(u_t)-\phi^*$ 和 $V(u_t):=V_{g,u_1,\beta}(u_t)=\langle\nabla\phi(u_t),u_t-v_t\rangle$, 代入不等式 (4.67) 得到

$$\begin{aligned}\lambda_{t+1}V(u_t)&\leqslant\phi(u_t)-\phi(u_{t+1})+\frac{\beta\lambda_{t+1}^2}{2}\|v_t-u_t\|^2\\&=\Delta_t-\Delta_{t+1}+\frac{\beta\lambda_{t+1}^2}{2}\|v_t-u_t\|^2.\end{aligned}\tag{4.71}$$

将上式两边同除以 Λ_{t+1} 并从 $t=1$ 加到某一给定的 t, 可得

$$\sum_{j=1}^{t}\frac{\lambda_{j+1}}{\Lambda_{j+1}}V(u_j)\leqslant\sum_{j=1}^{t}\left(\frac{\Delta_j}{\Lambda_{j+1}}-\frac{\Delta_{j+1}}{\Lambda_{j+1}}\right)+\sum_{j=1}^{t}\frac{\beta\lambda_{j+1}^2}{2\Lambda_{j+1}}\|v_j-u_j\|^2$$

$$= -\frac{\Delta_{t+1}}{\Lambda_{t+1}} + \sum_{j=2}^{t}\left(\frac{1}{\Lambda_{j+1}} - \frac{1}{\Lambda_j}\right)\Delta_j + \frac{\Delta_1}{\Lambda_2} + \sum_{j=1}^{t}\frac{\beta\lambda_{j+1}^2}{2\Lambda_{j+1}}\|v_j - u_j\|^2$$

$$\leqslant \sum_{j=1}^{t} j\Delta_j + t\beta D_X^2. \tag{4.72}$$

不等式 (4.72) 成立是因为 $\lambda_t := 2/t$ 和 $\Lambda_t = 2/[t(t-1)]$, 因此, $\Lambda_2 = 1$, $1/\Lambda_{j+1} - 1/\Lambda_j = j$. 由 (4.70), 可得

$$\Delta_j = \phi(u_j) - \phi^* \leqslant \frac{2\beta D_X^2}{t+1},$$

因此有

$$\sum_{j=1}^{t} j\Delta_j + t\beta D_X^2 \leqslant \frac{2\beta D_X^2}{t+1}\sum_{j=1}^{t} j + t\beta D_X^2 = 3t\beta D_X^2. \tag{4.73}$$

将不等式 (4.73) 代入 (4.72) 可以得到

$$\min_{j=1,\ldots,t} V(u_j)\sum_{j=1}^{t}\frac{\lambda_{j+1}}{\Lambda_{j+1}} \leqslant \sum_{j=1}^{t}\frac{\lambda_{j+1}}{\Lambda_{j+1}}V(u_j) \leqslant 3t\beta D_X^2. \tag{4.74}$$

注意到

$$\sum_{j=1}^{t}\frac{\lambda_{j+1}}{\Lambda_{j+1}} = \frac{t(t+1)}{2}, \tag{4.75}$$

代入 (4.74) 得

$$\min_{j=1,\ldots,t} V(u_j) \leqslant \frac{6\beta D_X^2}{t+1}. \tag{4.76}$$

考虑在外迭代第 k 步时, 令上面不等式右端小于给定 η_k, 我们可以得到子程序 CndG 的复杂度 T_k 为

$$T_k = \left\lceil \frac{6\beta_k D_X^2}{\eta_k} \right\rceil. \tag{4.77}$$

外迭代的复杂度:

对于外迭代, 其框架与加速梯度法相同. 由 (2.69) 可以得到

$$f(y_k) \leqslant (1-\gamma_k)f(y_{k-1}) + \gamma_k[f(z_k) + \langle\nabla f(z_k), x_k - z_k\rangle] + \frac{L\gamma_k^2}{2}\|x_k - x_{k-1}\|^2. \tag{4.78}$$

由于 x_k 是子问题 $\min_{x\in X}\{\langle\nabla f(z_k), x\rangle + \frac{\beta_k}{2}\|x - x_{k-1}\|_2^2\}$ 的 η_k 精度的近似解, 即最优性条件满足

$$\langle\nabla f(z_k) + \beta_k(x_k - x_{k-1}), x - x_k\rangle \geqslant \eta_k, \quad \forall\, x \in X, \tag{4.79}$$

等价于

$$\langle x_{k-1} - x_k, x_k - x\rangle \leqslant \frac{1}{\beta_k}\langle\nabla f(x_k), x - x_k\rangle + \frac{\eta_k}{\beta_k}. \tag{4.80}$$

利用上述不等式, 我们可以估计 x_k 与 x_{k-1} 的逼近程度,

$$\frac{1}{2}\|x_k - x_{k-1}\|^2 = \frac{1}{2}\|x_{k-1} - x\|^2 - \langle x_{k-1} - x_k, x_k - x\rangle - \frac{1}{2}\|x_k - x\|^2$$

$$\leqslant \frac{1}{2}\|x_{k-1} - x\|^2 + \frac{1}{\beta_k}\langle\nabla f(z_k), x - x_k\rangle - \frac{1}{2}\|x_k - x\|^2 + \frac{\eta_k}{\beta_k}. \tag{4.81}$$

上述不等式两边同乘以 $L\gamma_k^2$, 并注意到 $L\gamma_k \leqslant \beta_k$, 我们可以推出

$$\frac{L\gamma_k^2}{2}\|x_k - x_{k-1}\|^2 \leqslant \frac{L\gamma_k^2}{2}\|x_{k-1} - x\|^2 + \gamma_k\langle \nabla f(z_k), x - x_k\rangle - \frac{L\gamma_k^2}{2}\|x_k - x\|^2 + \gamma_k\eta_k. \tag{4.82}$$

将 (4.82) 代入 (4.78) 并利用 $f(x)$ 的凸性, 可以得到

$$f(y_k) - f(x) \leqslant (1-\gamma_k)[f(y_{k-1}) - f(x)] + \frac{\beta_k\gamma_k}{2}(\|x_{k-1} - x\|^2 - \|x_k - x\|^2) + \gamma_k\eta_k. \tag{4.83}$$

利用引理 2.1 得到收敛性估计的不等式

$$\begin{aligned} f(y_k) - f(x) \leqslant{}& \frac{\Gamma_k(1-\gamma_1)}{\Gamma_1}(f(y_0) - f(x)) \\ &+ \frac{\Gamma_k}{2}\sum_{i=1}^{k}\frac{\beta_i\gamma_i}{\Gamma_i}(\|x_{i-1} - x\|^2 - \|x_i - x\|^2) + \Gamma_k\sum_{i=1}^{k}\frac{\gamma_i\eta_i}{\Gamma_i}. \end{aligned} \tag{4.84}$$

令 $D_X = \sup_{x,y\in X}\|x - y\|$, $D_0 = \|x_0 - x^*\|^2$, 分析上面不等式, 注意到,

$$\begin{aligned} &\sum_{i=1}^{k}\frac{\beta_i\gamma_i}{\Gamma_i}(\|x_{i-1} - x\|^2 - \|x_i - x\|^2) \\ &= \frac{\beta_1\gamma_1}{\Gamma_1}\|x_0 - x\|^2 + \sum_{i=2}^{k}\left(\frac{\beta_i\gamma_i}{\Gamma_i} - \frac{\beta_{i-1}\gamma_{i-1}}{\Gamma_{i-1}}\right)\|x_{i-1} - x\|^2 - \beta_k\gamma_k\Gamma_k\|x_k - x\|^2 \\ &\leqslant \frac{\beta_1\gamma_1}{\Gamma_1}\|x_0 - x\|^2 + \sum_{i=2}^{k}\left(\frac{\beta_i\gamma_i}{\Gamma_i} - \frac{\beta_{i-1}\gamma_{i-1}}{\Gamma_{i-1}}\right)\cdot D_X^2 \\ &\leqslant \frac{\beta_k\gamma_k}{\Gamma_k}D_X^2. \end{aligned} \tag{4.85}$$

因此, 对任意序列 $\{\beta_k\}$、$\{\gamma_k\}$ 和 $\{\Gamma_k\}$ 满足 $L\gamma_k \leqslant \beta_k$ 时有

$$f(y_k) - f(x) \leqslant \frac{\beta_k\gamma_k}{2}D_X^2 + \Gamma_k\sum_{i=1}^{k}\frac{\gamma_i\eta_i}{\Gamma_i}. \tag{4.86}$$

若序列 $\{\beta_k\}$、$\{\gamma_k\}$ 和 $\{\Gamma_k\}$ 满足 $L\gamma_k \leqslant \beta_k$ 且

$$\frac{\beta_k\gamma_k}{\Gamma_k} \geqslant \frac{\beta_{k-1}\gamma_{k-1}}{\Gamma_{k-1}}, \quad \text{对任意 } k \geqslant 2, \tag{4.87}$$

则有

$$f(y_k) - f(x) \leqslant \Gamma_k\frac{\beta_1\gamma_1}{2}\|x_0 - x\|^2 + \Gamma_k\sum_{i=1}^{k}\frac{\gamma_i\eta_i}{\Gamma_i}. \tag{4.88}$$

综上, 我们可以通过构造序列 $\{\beta_k\}$、$\{\gamma_k\}$ 和 $\{\eta_k\}$ 来得到算法的收敛速率.

定理 4.6 若 $f \in C_L^{1,1}(\mathbb{R}^n)$ 且是凸函数, 则加速条件梯度法求解问题模型 2.4 的收敛速率如下.

(1) 取 $\beta_k = \frac{3L}{k+1}$、$\gamma_k = \frac{3}{k+2}$ 和 $\eta_k = \frac{LD_X^2}{k(k+1)}$ 时, 外迭代收敛速度为

$$f(y_k) - f(x^*) \leqslant \frac{15LD_X^2}{2(k+1)(k+2)}. \tag{4.89}$$

若外迭代总次数 ($\mathcal{FO}$ 的分析复杂度) 为 N, 则内迭代总次数 ($\mathcal{LO}$ 的分析复杂度) 上界为

$$\sum_{k=1}^{N} T_k \leqslant 9N^2 + 10N. \tag{4.90}$$

加速条件梯度法为了得到 ϵ 近似解所需要的 $\mathcal{FO}$ 和 $\mathcal{LO}$ 复杂度上界分别为

$$N(\mathcal{FO}) = \sqrt{\frac{15LD_X^2}{2\epsilon}}, \quad N(\mathcal{LO}) = \frac{135LD_X^2}{2\epsilon} + 10\sqrt{\frac{15LD_X^2}{2\epsilon}}. \tag{4.91}$$

(2) 令外迭代总次数 N 固定, 取 $\beta_k = \frac{2L}{k}$、$\gamma_k = \frac{2}{k+1}$ 和 $\eta_k = \frac{2LD_0^2}{Nk}$, 则外迭代收敛速度为

$$f(y_N) - f^* \leqslant \frac{6LD_0^2}{N(N+1)}. \tag{4.92}$$

内迭代总次数 ($\mathcal{LO}$ 的分析复杂度) 为

$$\sum_{k=1}^{N} T_k \leqslant \frac{6N^2 D_X^2}{D_0^2} + N. \tag{4.93}$$

加速条件梯度法为了得到 ϵ 近似解所需要的 $\mathcal{FO}$ 和 $\mathcal{LO}$ 复杂度上界分别为

$$N(\mathcal{FO}) = \mathcal{O}\left(D_0\sqrt{\frac{L}{\epsilon}}\right), \quad N(\mathcal{LO}) = \mathcal{O}\left(\frac{LD_X^2}{\epsilon} + D_0\sqrt{\frac{L}{\epsilon}}\right). \tag{4.94}$$

证明 对于定理的第一部分, 当取 $\beta_k = \frac{3L}{k+1}$、$\gamma_k = \frac{3}{k+2}$ 和 $\eta_k = \frac{LD_X^2}{k(k+1)}$ 时满足 $L\gamma_k \leqslant \beta_k$, 根据 Γ_k 的定义, 有

$$\Gamma_k = \frac{6}{k(k+1)(k+2)}, \quad \frac{\eta_i\gamma_i}{\Gamma_i} = 3LD_X^2. \tag{4.95}$$

因此, 由 (4.86), 有

$$f(y_k) - f(x^*) \leqslant \frac{9LD_X^2}{2(k+1)(k+2)} + \frac{6}{k(k+1)(k+2)}\sum_{i=1}^{k}\frac{\eta_i\gamma_i}{\Gamma_i} \leqslant \frac{15LD_X^2}{2(k+1)(k+2)}. \tag{4.96}$$

令不等式 (4.96) 右端小于 ϵ, 则可以得到子程序 $\mathcal{FO}$ 的分析复杂度为

$$N(\mathcal{FO}) = \sqrt{\frac{15LD_X^2}{2\epsilon}}. \tag{4.97}$$

若外迭代次数为 N, 则内迭代总次数的上界为

$$\sum_{k=1}^{N} T_k \leqslant \sum_{k=1}^{N}\left(\frac{6\beta_k D_X^2}{\eta_k} + 1\right) = 18\sum_{k=1}^{N} k + N = 9N^2 + 10N. \tag{4.98}$$

而子程序 $\mathcal{LO}$ 的分析复杂度为

$$N(\mathcal{LO}) = 9N^2(\mathcal{FO}) + 10N(\mathcal{FO}) = \frac{135LD_X^2}{2\epsilon} + 10\sqrt{\frac{15LD_X^2}{2\epsilon}}. \tag{4.99}$$

对于定理的第二部分, 当外迭代总次数 N 提前固定, 取 $\beta_k = \frac{2L}{k}$、$\gamma_k = \frac{2}{k+1}$ 和 $\eta_k = \frac{2LD_0^2}{Nk}$, 则

$$\Gamma_k = \frac{2}{k(k+1)}, \quad \frac{\gamma_k\eta_k}{\Gamma_k} = \frac{2LD_0^2}{N}.$$

于是, 由 (4.88) 外迭代收敛速度为

$$f(y_N) - f^* \leqslant \frac{\beta_1 \Gamma_N}{2}\|x_0 - x\|^2 + \Gamma_N \sum_{i=1}^{N} \frac{\gamma_i \eta_i}{\Gamma_i} \ = \ \frac{6LD_0^2}{N(N+1)}. \tag{4.100}$$

令不等式 (4.100) 右端小于 ϵ, 则可以得到子程序 $\mathcal{FO}$ 的分析复杂度为

$$N(\mathcal{FO}) = \mathcal{O}\left(D_0\sqrt{\frac{L}{\epsilon}}\right). \tag{4.101}$$

由 (4.77) 得到内迭代总次数 ($\mathcal{LO}$ 的分析复杂度) 为

$$\sum_{k=1}^{N} T_k \leqslant \sum_{k=1}^{N}\left(\frac{6\beta_k D_X^2}{\eta_k} + 1\right) = \frac{6N^2 D_X^2}{D_0^2} + N. \tag{4.102}$$

因此, 子程序 $\mathcal{LO}$ 的分析复杂度为

$$N(\mathcal{LO}) = \frac{6D_X^2}{D_0^2} N^2(\mathcal{FO}) + N(\mathcal{FO}) = \mathcal{O}\left(\frac{LD_X^2}{\epsilon} + D_0\sqrt{\frac{L}{\epsilon}}\right). \tag{4.103}$$

证毕. □

由上述定理可以看出, 加速条件梯度法的 $\mathcal{FO}$ 和 $\mathcal{LO}$ 分析复杂度分别达到了函数集合 $C_L^{1,1}(X)$ 对应的复杂度下界, 即 $\mathcal{O}(1/\sqrt{\epsilon})$ 和 $\mathcal{O}(1/\epsilon)$.

5 随机优化算法的复杂度分析

5.1 引言

本节研究随机优化问题的复杂度分析. 首先, 定义随机优化问题的具体形式. 考虑随机优化问题

$$\min_{x\in X}\{f(x) := \mathbb{E}_\xi[F(x,\xi)]\}, \tag{5.1}$$

其中 $X \subset \mathbb{R}^n$ 是一个非空有界闭凸集合; ξ 是一个随机向量, 其分布 P 的支撑集满足 $\Xi \subset \mathbb{R}^d$. 定义函数 $F:\ X \times \Xi \ \to \ \mathbb{R}$, 对任意 $\xi \in \Xi$, $F(x,\xi)$ 都是凸函数, 且关于 ξ 的期望

$$\mathbb{E}[F(x,\xi)] = \int_\Xi F(x,\xi) dP(\xi) \tag{5.2}$$

是有意义的, 且对任意 $x \in X$ 都为有限值. 因此, 我们可以推出 $f(\cdot)$ 在 X 上是凸的并且具有有限值. 同样, 由凸函数在有界闭集上连续, 我们推出 $f(\cdot)$ 在 X 上连续.

有了这些假设, (5.1) 变成了一个确定的凸优化问题. 但在求解优化问题 (5.1) 前, 我们需要首先计算出期望 (5.2) 的具体形式. 从计算复杂度角度讲, 即使 ξ 的分布 P 已知, 随着维数的增加, (5.2) 的计算也会变得很困难. 事实上, 对任意 $x \in X$, 我们需要计算 $f(x)$ 的一个近似逼近值 $\hat{f}(x)$, 使得 $|\hat{f}(x) - f(x)| \leqslant \epsilon$, 而这需要在不同的 $\xi \in \Xi$ 处计算 $\mathcal{O}(\frac{1}{\epsilon^m})$ 次 $f(x,\xi)$. 因此在高维情形下, 直接求解优化问题 (5.1) 很困难, 我们需要采用随机算法, 例如, Monte Carlo 抽样技巧来降低求解 (5.1) 的复杂度. 为此, 我们需要对问题 (5.1) 做如下假设:

(A1) 对于随机向量 ξ, 存在已知的 Monte Carlo 策略, 能够产生一系列独立同分布的样本 $\{\xi_i\}_{i=1}^N$.

对于随机优化问题 (5.1), 我们给出随机向量 ξ 分布已知和未知情形下的两个常见例子: 随机效用模型 (stochastic utility model) 和机器学习模型.

例 5.1 (随机效用模型) 常见的形式是

$$\min_{x\in X}\left\{f(x)=\mathbb{E}_\xi\left[\phi\Big(\sum_{i=1}^n\Big(\frac{i}{n}+\xi_i\Big)x_i\Big)\right]\right\},\tag{5.3}$$

其中

$$X=\left\{x\in\mathbb{R}^n:x\geqslant 0,\sum_{i=1}^n x_i=1\right\},$$

ξ_i 独立且 $\xi_i\sim N(0,1)$, $\phi(\cdot)$ 是逐段线性的凸函数,

$$\phi(t)=\max\{a_1t+b_1,\ldots,a_mt+b_m\},\tag{5.4}$$

其中 a_i 和 b_i 是已知常数.

例 5.2 (机器学习模型) 对于机器学习中的分类或回归问题, 我们有 n 个训练样本 $\{(x_1,y_1),\ldots,(x_n,y_n)\}$, 其中样本特征 $x_i\in\mathcal{X}$, 样本标签 $y_i\in\mathcal{Y}$. 假设其中的样本 (x_i,y_i) 都是从某一未知分布 D 中独立抽样出来. 我们需要求解分类或回归模型 $h(\,\cdot\,;\omega):\mathcal{X}\to\mathcal{Y}$, 其中 ω 是预测函数的参数. 为此, 我们构造误差函数 $\ell(h(\,\cdot\,;\omega),\,\cdot\,):\mathcal{X}\times\mathcal{Y}\to\mathbb{R}$, 并定义其在分布 D 上的期望风险 (expected risk)

$$R(w)=\mathbb{E}_{(x,y)\sim D}[\ell(h(x;\omega),y)].\tag{5.5}$$

通过求解优化问题

$$\min_w\ R(w)+r(w)\tag{5.6}$$

求得分类或回归模型 $h(x;\omega)$ 的最优参数 w^*, 其中 $r(\cdot)$ 是正则项, 可以取 $r(w)=\|w\|_2^2$ 或 $r(w)=\|w\|_1$, 用来控制模型 $h(\,\cdot\,;\omega)$ 的复杂度, 防止过拟合.

对例 5.2 来说, 由于数据分布 D 未知, 期望风险 (5.5) 无法计算, 但考虑到数据集 $(x_i,y_i)\sim D$ 独立同分布, 因此, 我们用经验风险 (empirical risk)

$$R_n(w):=\frac{1}{n}\sum_{i=1}^n\ell(h(x_i;\omega),y_i)\tag{5.7}$$

近似 $R(w)$, 通过求解优化问题

$$\min_w\ R_n(w)+r(w)\tag{5.8}$$

得到 $h(x;\omega^*)$ 的一个近似. 这样会出现一个问题, 我们如何衡量优化问题 (5.8) 的解与优化问题 (5.6) 解之间的关系?

这时, 我们需要首先定义随机优化问题 (5.1) 解的概念. 如果将随机优化问题 (5.1) 看成是一个确定性问题, 先计算出 (5.1) 期望的具体形式, 再用确定性优化算法 (如梯度法和 Newton 法) 求解其 ϵ 近似解, 即

$$\text{寻找 }\bar{x}\in X:\ f(\bar{x})-f^*\leqslant\epsilon.\tag{5.9}$$

然而计算 (5.1) 的具体形式非常困难. 因此, 为了绕过积分计算, 我们需要采用一些随机算法求解 (5.1). 由假设 (A1), 对随机变量 ξ 采样得到样本数据 $\{\xi_i\}$, 通过处理 $\{f(x,\xi_i)\}$, 我们可以得到 (5.1) 的一个近似解 $\tilde{x}$. 注意到近似解 $\tilde{x}$ 是一个关于 ξ 的函数, 因此, 它是一个随机变量. 传统确定性优化问题的解概念 (5.9) 不再适应于 $\tilde{x}$. 我们可以引入如下解概念.

定义 5.1 由随机优化算法产生的随机变量 $\tilde{x} \in X$, 如果其满足

$$\mathbb{E}_{\tilde{x}}[f(\tilde{x}) - f^*] \leqslant \epsilon, \tag{5.10}$$

则称其为随机优化问题 (5.1) 的 ϵ 近似解.

由于同样无法精确计算期望, 解概念 (5.10) 与 (5.9) 并没有很大的区别. 因此, 我们可以弱化用期望来衡量随机变量解 $\tilde{x}$ 的精确性, 使用关于解不等式成立的概率置信水平来衡量其解的好坏, 即引入 (ϵ, δ) 近似解.

定义 5.2 由随机优化算法产生的随机变量 $\tilde{x} \in X$, 如果其满足

$$\mathbf{Prob}(f(\tilde{x}) - f^* \geqslant \epsilon) \leqslant \delta, \tag{5.11}$$

则称其为随机优化问题 (5.1) 的 (ϵ, δ) 近似解.

在随机优化问题的 (ϵ, δ) 近似解概念中, δ 代表解收敛到 ϵ 精度的置信水平. 当 $\delta = 0$ 时, 近似解 (5.11) 等价于解 (5.9). 当 δ 越趋近于 0, 表示随机算法得到的近似解 $\tilde{x}$ 收敛到 ϵ 精度的可能性越高 (置信水平越高).

有了上述随机优化问题的解概念, 我们可以研究随机优化算法的收敛性和复杂度. 本节的结构如下. 我们先介绍求解随机优化问题的两个策略: 采样平均逼近和随机逼近算法, 并比较它们收敛性的置信水平; 然后介绍随机梯度法在非光滑问题中的复杂度; 最后介绍随机梯度法在光滑凸问题和光滑非凸问题中的复杂度分析.

5.2 随机优化算法

本小节介绍 3 种不同的随机优化算法: 采样平均逼近算法 (sample average approximation method)、随机逼近算法 (stochastic approximation method) 和集成随机逼近算法 (ensemble stochastic approximation method). 其中采样平均逼近算法首先通过采样 (如 Monte Carlo 算法) 对目标函数进行近似, 然后求解近似函数的最优值, 更详细的可以参见文献 [32–36]. 随机逼近算法的一个典型例子是随机梯度法, 它直接利用目标函数的随机梯度信息来更新迭代点求解最优近似解. 随机逼近算法的历史可以追溯到 1951 年 Robbins 和 Monro 的开创性工作 [37], 此后随机逼近算法 (随机梯度法) 被广泛应用于求解随机优化问题, 可参见文献 [38–40]. 随着随机梯度法在机器学习问题中的广泛应用, 产生了一些新形式的随机梯度法, 具体可以参见文献 [15–18,39,41–46]. 集成随机逼近算法是利用机器学习中集成算法的思想, 将多次随机逼近算法的解综合起来得到置信水平最高的近似解, 具体的可参见文献 [47].

5.2.1 采样平均逼近算法的复杂度分析

在求解随机优化问题 (5.1) 时, 采样平均逼近算法利用假设 (A1) 产生样本序列 $\{\xi_i\}_{i=1}^N$, 构造目标函数 (5.1) 的逼近,

$$\hat{f}_N(x) = \frac{1}{N}\sum_{i=1}^{N} F(x, \xi_i). \tag{5.12}$$

然后再用确定的优化算法 (如梯度法和 Newton 法) 求 $\hat{f}_N(x)$ 在 X 上的极小值. 例如, 在例 5.2 中, 我们采用经验风险 $R_n(w)$ 来逼近期望风险 $R(w)$. 采样平均逼近算法 (算法 5.1) 具体如下.

算法 5.1 采样平均逼近算法

1. 对随机变量 ξ 进行抽样, 产生样本序列 ξ_i, $i=1,\dots,N$;
2. 构造 $f(x)$ 的逼近函数:

$$\hat{f}_N(x)=\frac{1}{N}\sum_{i=1}^{N}F(x,\xi_i); \tag{5.13}$$

3. 求解确定优化问题:

$$\min_{x\in X}\left\{\hat{f}_N(x)=\frac{1}{N}\sum_{i=1}^{N}F(x,\xi_i)\right\}. \tag{5.14}$$

当采样的样本总量 N 变大时, 对任意的 $x\in X$ 和 $\epsilon>0$, 有结论

$$\mathbf{Prob}\left(\left|\frac{1}{N}\sum_{i=1}^{N}F(x,\xi_i)-f(x)\right|\geqslant\epsilon\right)\to 0. \tag{5.15}$$

从计算角度来讲, 上述结论非常弱, 因为当 ξ 的维数很高时在某一确定的点 x 处对 (5.1) 中 $f(x)$ 做逼近会很困难. 但是, 当 $f(x,\xi)$ 关于 ξ 一致有界时, 我们可以得到比 (5.14) 更强的结论: 可以设计在多项式时间内逼近 $f(x)$ 的方法. 我们可以衡量 $\hat{f}(x)$ 逼近 $f(x)$ 的复杂度.

定理 5.1 假定 (5.1) 中函数 $F(x,\xi)$ 任意变差有界, 即存在 $V<\infty$, 其中

$$V=\max\{F(x_1,\xi_1)-F(x_2,\xi_2):x_1,x_2\in X,\ \xi_1,\xi_1\in\Xi\}, \tag{5.16}$$

则对任意 $\epsilon>0$ 和 $\delta\in(0,1)$, 当抽样样本数量取 $N=\lceil\frac{V^2}{2\epsilon^2}\log(\frac{2}{\delta})\rceil$ 时, 有

$$\mathbf{Prob}\{|\hat{f}_N(x)-f(x)|>\epsilon\}\leqslant\delta. \tag{5.17}$$

为了证明定理 5.1, 我们需要介绍随机优化算法收敛性证明的常用概率分析工具: Hoeffding 不等式.

引理 5.1 (Hoeffding 不等式) 设 $Z_1,\dots,Z_N$ 是具有相同期望 $\mu>0$、相互独立的随机变量, 且 $\mathbf{Prob}(0\leqslant Z_i\leqslant V)=1$, $i=1,\dots,N$, 则对于 $\bar{Z}_N=\frac{Z_1+\cdots+Z_N}{N}$, 有

$$\mathbf{Prob}(\bar{Z}_N-\mu\geqslant\epsilon)\leqslant\exp\left(-\frac{2N\epsilon^2}{V^2}\right). \tag{5.18}$$

有了引理 5.1 的不等式工具, 我们可以证明定理 5.1 的收敛性结果.

证明 令 $\underline{f}=\inf_{x,\xi}F(x,\xi)$, 定义随机变量 $Z(\xi)=F(x,\xi)-\underline{f}$ 和 $W(\xi)=V-Z(\xi)$, 我们有 $0\leqslant Z(\xi)\leqslant V$ 和 $0\leqslant W(\xi)\leqslant V$, 同时,

$$\mu=\mathbb{E}(Z)=f(x)-\underline{f}\quad\text{和}\quad\mu'=\mathbb{E}(W)=V-\mu. \tag{5.19}$$

注意到,

$$\begin{aligned}\mathbf{Prob}(|\bar{Z}_N-\mu|)&=\mathbf{Prob}(\bar{Z}_N-\mu\geqslant\epsilon)+\mathbf{Prob}(\bar{Z}_N-\mu\leqslant-\epsilon)\\&=\mathbf{Prob}(\bar{Z}_N-\mu\geqslant\epsilon)+\mathbf{Prob}(\bar{W}_N-\mu'\geqslant\epsilon).\end{aligned} \tag{5.20}$$

对 Z 和 W 利用引理 5.1, 并取 $N=\lceil\frac{V^2}{2\epsilon^2}\log(\frac{2}{\delta})\rceil$, 有

$$\mathbf{Prob}\{|\hat{f}_N(x)-f(x)|>\epsilon\}\leqslant 2\exp\left(-\frac{2N\epsilon^2}{V^2}\right)\leqslant\delta. \tag{5.21}$$

证毕. □

由定理 5.1 可知, 对任意 $x \in X$, 为了对 (5.1) 中 $f(x)$ 做 ϵ 精度的逼近, 且逼近的置信水平为 δ, 我们需要至少对随机变量抽样 $\lceil \frac{V^2}{2\epsilon^2} \log(\frac{2}{\delta}) \rceil$ 次. 设 x^* 表示优化问题 (5.1) 的最优解, $\hat{x}^*$ 表示优化问题 (5.14) 的最优解. 定理 5.1 告诉我们,

$$\mathbf{Prob}\{|\hat{f}_N(x^*) - f(x^*)| > \epsilon\} \leqslant \delta. \tag{5.22}$$

而我们更关注是否有

$$\mathbf{Prob}\{|\hat{f}_N(\hat{x}^*) - f(x^*)| > \epsilon\} \leqslant \delta \tag{5.23}$$

成立. 因为采样平均逼近算法实际中求解的是优化问题 (5.14). 另外, 考虑到数值优化算法只能对 $\hat{x}^*$ 求解到某一 ϵ 精度, 我们需要知道 $\hat{x}^*$ 的 ϵ 近似解是否也是 x^* 的 (ϵ, δ) 近似解? Shapiro 和 Nemirovski 在文献 [48] 中给出了如下定理.

定理 5.2 若 $X \subset \mathbb{R}^n$, 令 $D := \sup_{x,y \in X} \|x - y\|$, 假设存在 $L > 0$, 使得对任意 $x, y \in X$ 和 $\xi \in \Xi$, 有 $|F(x, \xi) - F(y, \xi)| \leqslant L\|x - y\|$. 如果 (5.14) 中 N 满足

$$N \geqslant \mathcal{O}(1)\left(\frac{DL}{\epsilon}\right)^2 \left[n \ln\left(\frac{DL}{\epsilon}\right) + \ln\left(\frac{1}{\delta}\right)\right], \tag{5.24}$$

则优化问题 (5.14) 每个精度为 $\epsilon/2$ 的近似解都是随机优化问题 (5.1) 的 (ϵ, δ) 近似解.

有了上述定理 5.2, 我们就可以分析抽样平均逼近算法的复杂度. 考虑为求得随机优化问题 (5.1) 的 (ϵ, δ) 近似解需要对 $F(x, \xi)$ 求导多少次. 对 $\hat{f}_N(x)$ 求一次导数需要求 N 次 $F(x, \xi)$ 的导数. 于是有如下复杂度结论:

(1) 若 $\hat{f}_N(x) \in C^{0,1}_L(X)$ 且是凸函数, 利用投影次梯度法求 (5.14) 精度为 $\epsilon/2$ 的近似解需要对 $\hat{f}_N(x)$ 求 $\mathcal{O}(L^2D^2/\epsilon^2)$ 次导数, 若使该解为 (5.1) 的 (ϵ, δ) 近似解, 则总共需要对 $F(x, \xi)$ 求导的次数不少于

$$\mathcal{O}\left(\frac{D^4L^4}{\epsilon^4}\left[n \ln\left(\frac{DL}{\epsilon}\right) + \ln\left(\frac{1}{\delta}\right)\right]\right). \tag{5.25}$$

忽略常数项后, 抽样平均逼近算法求解随机优化问题 (5.1) 的复杂度为

$$\mathcal{O}\left(\frac{n}{\epsilon^4} \ln\frac{1}{\epsilon} + \frac{1}{\epsilon^4} \ln\frac{1}{\delta}\right). \tag{5.26}$$

(2) 若 $\hat{f}_N(x) \in C^{1,1}_L(X)$ 且是凸函数, 利用加速投影次梯度法求 (5.14) 精度为 $\epsilon/2$ 的近似解需要对 $\hat{f}_N(x)$ 求 $\mathcal{O}(L^{\frac{1}{2}}D/\epsilon^{\frac{1}{2}})$ 次导数, 若使该解为 (5.1) 的 (ϵ, δ) 近似解, 则总共需要对 $F(x, \xi)$ 求导的次数不少于

$$\mathcal{O}\left(\frac{D^3L^{2.5}}{\epsilon^{2.5}}\left[n \ln\left(\frac{DL}{\epsilon}\right) + \ln\left(\frac{1}{\delta}\right)\right]\right). \tag{5.27}$$

忽略常数项后, 抽样平均逼近算法求解随机优化问题 (5.1) 的复杂度为

$$\mathcal{O}\left(\frac{n}{\epsilon^{2.5}} \ln\frac{1}{\epsilon} + \frac{1}{\epsilon^{2.5}} \ln\frac{1}{\delta}\right). \tag{5.28}$$

5.2.2 随机逼近算法的复杂度分析

随机逼近算法是随机梯度算法的原型, 它与采样平均逼近算法的区别在于每次迭代不用计算 N 次 $F(x, \xi)$ 的导数, 而是利用产生一个随机梯度来近似 $f(x)$ 的梯度. 例如, 当 (5.1) 的目标函数 $f(x)$ 非光滑时, 随机逼近算法利用如下假设.

(A2) 存在子程序 $\mathcal{SFO}$ 对任意 $x\in X$ 和随机样本 $\xi\in\Xi$, 返回函数值 $F(x,\xi)$ 和一阶随机次梯度 $G(x,\xi)$, 满足

$$\mathbb{E}_\xi[G(x,\xi)]=g(x)\in\partial f(x). \tag{5.29}$$

在假设 (A2) 下, 随机逼近算法 (算法 5.2) 通过调用子程序 $\mathcal{SFO}$ 产生随机次梯度 $G(x,\xi)$ 信息, 每次迭代只有一次次梯度计算过程, 而采样平均逼近算法需要计算 N 次.

算法 5.2 随机逼近算法框架 $\mathcal{S}$

输入: $x_0\in X$, 初始信息集合 $I_{-1}=\emptyset$.
对 $k=0,1,\ldots,N$ 执行迭代,
1. 根据 ξ 的分布生成样本 ξ_k;
2. 在 ξ_k 处调用子程序 $\mathcal{SFO}$, 更新信息集合 $I_{k+1}=I_k\cup(x_k,\xi_k,\mathcal{SFO}(x_k,\xi_k))$;
3. 根据信息集合 I_{k+1} 更新迭代点 x_{k+1}.
输出: $\bar{x}=\mathcal{S}(x_0)$.

利用机器学习中集成学习的概念, 我们可以对随机逼近算法进行集成, 得到集成随机逼近算法 (算法 5.3). 该算法最早被 Nesterov 和 Vial [47] 提出.

算法 5.3 集成随机逼近算法 $\mathcal{M}$

输入: $x_0\in\mathbb{R}^n$, 随机逼近基算法 $\mathcal{S}$, 每个基算法迭代次数 N, 集成次数 K.
对 $j=1,\ldots,K$ 执行迭代,
1. 调用随机逼近基算法 $\mathcal{S}$, 得到近似解 $\bar{x}_j=\mathcal{S}(x_0)$.
输出: $\hat{x}=\mathcal{M}(x_0)=\frac{1}{K}\sum_{j=1}^K\bar{x}_j$.

我们可以研究算法 5.3 的解 $\hat{x}=\mathcal{M}(x_0)$ 与算法 5.2 的解 $\bar{x}=\mathcal{S}(x_0)$ 之间的关系, 并比较它们求 (ϵ,δ) 近似解的复杂度. 为了分析复杂度, 我们对 (5.1) 的目标函数做如下假设:

(1) $f(x)$ 在 X 上的总变差有界, 即 $V_f=\sup\{f(x)-f^*:x\in X\}<\infty$;

(2) $F(x,\xi)$ 在 $X\times\Xi$ 上的任意变差有界.

对算法 5.2 的解 $\bar{x}$, 若我们可以证明存在关于迭代次数 N 的单调递减函数 $C_{\mathcal{S}}(N)$, 使得

$$\mathbb{E}\,[f(\bar{x})-f^*]\leqslant C_{\mathcal{S}}(N), \tag{5.30}$$

则当 N 满足 $C_{\mathcal{S}}(N)\leqslant\epsilon\delta$ 时, $\bar{x}$ 是 (5.1) 的 (ϵ,δ) 近似解. 因此由 Markov 不等式, 有

$$\mathbf{Prob}\{f(\bar{x})-f^*\geqslant\epsilon\}\leqslant\frac{\mathbb{E}[f(\bar{x})-f^*]}{\epsilon}\leqslant\frac{C_{\mathcal{M}}(N)}{\epsilon}\leqslant\delta. \tag{5.31}$$

而对于算法 5.3 的解 $\hat{x}$, 我们有如下引理.

引理 5.2 设 $\hat{x}$ 为算法 5.3 的解, 则对任意 $\beta>0$, 有

$$\mathbf{Prob}\{f(\hat{x})-f^*\geqslant C_{\mathcal{M}}(N)+\beta\}\leqslant\exp\left\{-\frac{2K\beta^2}{V_f^2}\right\}. \tag{5.32}$$

证明 利用 $f(x)$ 的凸性得到 $f(\hat{x})\leqslant\frac{1}{K}\sum_{j=1}^K f(\bar{x}_j)$, 于是,

$$\mathbf{Prob}\{f(\hat{x})-f^*\geqslant C_{\mathcal{M}}(N)+\beta\}\leqslant\mathbf{Prob}\left\{\frac{1}{K}\sum_{j=1}^K f(\bar{x}_j)-f^*\geqslant C_{\mathcal{M}}(N)+\beta\right\}. \tag{5.33}$$

令 $Z = f(\bar{x}) - f^*$, $\bar{Z}_K = \frac{1}{K}\sum_{j=1}^{K} f(\bar{x}_j) - f^*$, $\mu = \mathbb{E}[f(\bar{x}) - f^*] = \mathbb{E}(Z)$, 由假设有 $0 \leqslant Z \leqslant V_f$. 对 Z 利用 Hoeffding 不等式, 可得

$$\begin{aligned}
&\mathbf{Prob}\left\{\frac{1}{K}\sum_{j=1}^{K} f(\bar{x}_j) - f^* \geqslant C_{\mathcal{M}}(N) + \beta\right\} \\
&\quad = \mathbf{Prob}\{\bar{Z}_K \geqslant \mathbb{E}[f(\bar{x}) - f^*] + \beta\} \leqslant \mathbf{Prob}\{\bar{Z}_K \geqslant \mu + \beta\} \\
&\quad \leqslant \exp\left\{\frac{-2K\beta^2}{V_f^2}\right\}. \qquad (5.34)
\end{aligned}$$

于是得到不等式 (5.32). □

定理 5.3 令 N 和 K 按如下方式选取:

$$N = \left\lceil C_{\mathcal{S}}^{-1}\left(\frac{\epsilon V_f}{2}\right)\right\rceil, \quad K = \left\lceil \frac{2}{\epsilon^2}\ln\frac{1}{\delta}\right\rceil, \qquad (5.35)$$

则算法 5.3 的解 $\hat{x}$ 是随机优化问题 (5.1) 的 $(\epsilon V_f, \delta)$ 近似解, 即

$$\mathbf{Prob}\{f(\hat{x}) - f^* \geqslant \epsilon V_f\} \leqslant \delta. \qquad (5.36)$$

且算法 5.3 总共需要调用子程序 $\mathcal{SFO}$ 的次数, 即计算复杂度 T 满足

$$T = K \cdot N \leqslant \left(1 + C_{\mathcal{S}}^{-1}\left(\frac{\epsilon V_f}{2}\right)\right) \cdot \left(1 + \frac{2}{\epsilon^2}\ln\frac{1}{\delta}\right). \qquad (5.37)$$

证明 令 $\beta = \frac{1}{2}\epsilon V_f$, 根据引理 5.2, 将 N 和 K 的取值 (5.35) 代入 (5.32) 即得结论. □

由随机梯度法的收敛性结论:

$$\mathbb{E}[f(\bar{x}_1^k) - f^*] \leqslant \frac{DM}{\sqrt{k}}, \qquad (5.38)$$

因此有 $C_{\mathcal{S}}(N) = DM/\sqrt{N}$. 由 (5.31), 令 $C_{\mathcal{S}}(N) \leqslant \epsilon V_f \delta$, 则为了得到 $(\epsilon V_f, \delta)$ 近似解, 算法 5.2 迭代次数 N 需要满足

$$N = \mathcal{O}\left[\frac{1}{\epsilon^2\delta^2}\left(\frac{DM}{V_f}\right)^2\right]. \qquad (5.39)$$

若基算法 $\mathcal{S}$ (算法 5.2) 是随机梯度法, 则可以考虑算法 $\mathcal{M}$ (算法 5.3) 求解 $(\epsilon V_f, \delta)$ 近似解的复杂度, 由定理 5.35 得

$$N = \mathcal{O}\left[\frac{1}{\epsilon^2}\left(\frac{MR}{V_f}\right)^2\right], \quad T = K \cdot N = \mathcal{O}\left[\frac{1}{\epsilon^4}\ln\frac{1}{\delta}\left(\frac{MR}{V_f}\right)^2\right]. \qquad (5.40)$$

我们将算法 $\mathcal{S}$ 与 $\mathcal{M}$ 的复杂度结论总结到表 2.

表 2 随机次梯度法与集成随机次梯度法计算复杂度的比较

	算法 $\mathcal{S}$	算法 $\mathcal{M}$
基算法个数 K	1	$\frac{1}{\epsilon^2}\ln\frac{1}{\delta}$
基算法迭代次数 N	$\frac{1}{\epsilon^2\delta^2}(\frac{MR}{V_f})^2$	$\frac{1}{\epsilon^2}(\frac{MR}{V_f})^2$
总计算复杂度	$\frac{1}{\epsilon^2\delta^2}(\frac{MR}{V_f})^2$	$\frac{1}{\epsilon^4}\ln\frac{1}{\delta}(\frac{MR}{V_f})^2$

5.3 非光滑随机优化问题

与梯度法和镜像下降法类似, 我们可以推广得到随机逼近算法框架下的随机梯度法 (算法 5.4) 和随机镜像下降法 (算法 5.5).

算法 5.4 随机梯度法 (stochastic gradient method)

输入: $x_0 \in \mathbb{R}^n$ 和步长序列 $\{\gamma_k\}$.

对 $k = 0, 1, \ldots$ 执行迭代,

1. 生成样本 ξ_k;

2. 在 (x_k, ξ_k) 处调用子程序 $\mathcal{SFO}$ 得到随机梯度 $G(x_k, \xi_k)$, 并更新

$$x_{k+1} = \operatorname*{argmin}_{x \in X} \|x_k - \gamma_k G(x_k, \xi_k)\|_2. \tag{5.41}$$

算法 5.5 随机镜像下降法 (stochastic mirror descent method)

输入: $x_0 \in \mathbb{R}^n$, 步长序列 $\{\gamma_k\}$, Bregman 距离 $V(x, y)$.

对 $k = 0, 1, \ldots$ 执行迭代,

(1) 根据 ξ 分布, 生成样本 ξ_k,

(2) 在 (x_k, ξ_k) 处调用子程序 $\mathcal{SFO}$ 得到随机梯度 $G(x_k, \xi_k)$, 并更新

$$x_{k+1} = \operatorname*{argmin}_{x \in X} \ \gamma_k \langle G(x_k, \xi_k), x\rangle + V(x_k, x). \tag{5.42}$$

当取 Bregman 距离为 $V(x, y) = \frac{1}{2}\|x-y\|^2$ 时, 随机梯度法 (算法 5.4) 与随机镜像下降法 (算法 5.5) 等价. 因此, 我们考虑随机镜像下降法的复杂度分析. 考虑随机优化问题 (5.1) 在下列随机梯度假设时的 ϵ 近似解和 (ϵ, δ) 近似解的复杂度. 假设 $\{\xi_t\}$ 是关于 ξ 的独立同分布随机变量序列, 考虑对任意的 $x \in X$, 存在常数 $M_* > 0$, 条件 (A2) 产生的随机次梯度 $G(x, \xi_t)$ 满足下列条件之一:

(A3) 存在 $M_* > 0$, 使得 $\|G(x, \xi_t)\|_* \leqslant M_*$;

(A4) 存在 $M_* > 0$, 使得 $\mathbb{E}_{\xi_t}[\exp(\frac{\|G(x,\xi_t)\|_*^2}{M_*^2})] \leqslant \exp(1)$;

(A5) 存在 $M_* > 0$, 使得 $\mathbb{E}_{\xi_t}[\|G(x, \xi_t)\|_*^2] \leqslant M_*^2$.

不难看出上述假设的包含关系如下:

$$\text{(A3)} \ \Rightarrow \ \text{(A4)} \ \Rightarrow \ \text{(A5)}. \tag{5.43}$$

假设条件 (A3) 最强, (A5) 最弱. 因此, 我们先考虑在假设条件 (A5) 下 ϵ 近似解的复杂度分析.

定理 5.4 若假设条件 (A1)、(A2) 和 (A5) 成立, 令 $\bar{x}_s^k = (\sum_{t=s}^k \gamma_t x_t)/(\sum_{t=s}^k \gamma_t)$, 其中 x_t 表示随机镜像下降法 (算法 5.5) 的迭代点, 则有如下收敛性结果:

$$\mathbb{E}[f(\bar{x}_s^k)] - f^* \leqslant \left(\sum_{t=s}^k \gamma_t\right)^{-1} \left[\mathbb{E}[V(x_s, x^*)] + (2\nu)^{-1} M_*^2 \sum_{t=s}^k \gamma_t^2\right]. \tag{5.44}$$

证明 记 $g_t := g(x_t) \in \partial f(x_t)$, $G_t := G(x_t, \xi_t) \in \partial_x F(x_t, \xi_t)$, 令 $\Delta_t = g_t - G_t$, $t = 1, \ldots, k$. 由 f 的凸性和引理 3.2, 有

$$\begin{aligned}\gamma_t[f(x_t) - f(x)] &\leqslant \gamma_t\langle G_t, x_t - x\rangle + \gamma_t\langle \Delta_t, x_t - x\rangle \\ &\leqslant V(x_t, x) - V(x_{t+1}) + \gamma_t\langle G_t, x_t - x_{t+1}\rangle - V(x_t, x_{t+1}) + \gamma_t\langle \Delta_t, x_t - x\rangle\end{aligned}$$

$$\leqslant V(x_t,x)-V(x_{t+1})+\nu^{-1}\gamma_t^2\|G_t\|^2+\gamma_t\langle\Delta_t,x_t-x\rangle, \tag{5.45}$$

其中 (5.45) 成立利用了 Cauchy-Schwarz 不等式以及对任意 $a>0$ 不等式 $bt-at^2/2\leqslant b^2/(2a)$ 都成立的事实. 对上述不等式左右两边取期望, 考虑到假设 (A2) 和 (A5), 有

$$\gamma_t\mathbb{E}[f(x_t)-f(x)]\leqslant\mathbb{E}[V(x_t,x)-V(x_{t+1},x)]+(2\nu)^{-1}\gamma_t^2M^2. \tag{5.46}$$

将上述不等式从 $t=s$ 到 $t=k$ 求和即得结论 (5.44). □

推论 5.1 对随机镜像下降法 (算法 5.5) 取固定步长 $\gamma_t=\frac{\sqrt{2\nu}D_{w,X}}{M\sqrt{k}}$, 则有收敛性结果

$$\mathbb{E}[f(\bar{x}_1^k)-f^*]\leqslant D_{w,X}M\sqrt{\frac{2}{\nu k}}, \tag{5.47}$$

其 ϵ 近似解的复杂度为

$$\mathcal{O}\left(\frac{D_{w,X}^2M^2}{\epsilon^2}\right). \tag{5.48}$$

定理 5.5 若假设条件 (A1)、(A2) 和 (A5) 成立, 则随机镜像下降法 (算法 5.5) 求得 (ϵ,δ) 近似解的复杂度为

$$\mathcal{O}\left(\frac{1}{\epsilon^2\delta^2}\right). \tag{5.49}$$

上述定理的证明可以参见文献 [6, 44].

定理 5.6 若假设条件 (A1)、(A2) 和 (A4) 成立, 则随机镜像下降法 (算法 5.5) 求得 (ϵ,δ) 近似解的复杂度为

$$\mathcal{O}\left(\frac{1}{\epsilon^2}\ln^2\frac{1}{\delta}\right). \tag{5.50}$$

证明 记 $g(x_t):=\mathbb{E}_\xi[F(x_t,\xi)]\in\partial f(x_t)$, 注意到

$$\|g(x_t)\|_*=\|\mathbb{E}(G(x_t,\xi_t)|\xi_{[t-1]})\|_*\leqslant\sqrt{\mathbb{E}(\|G(x_t,\xi_t)\|_*^2|\xi_{[t-1]})}\leqslant M_*. \tag{5.51}$$

记 $\Delta_t=g(x_t)-G(x_t,\xi_t)$, 由于

$$\|\Delta_t\|_*^2=\|g(x_t)-G(x_t,\xi_t)\|_*^2\leqslant(\|G(x_t,\xi_t)\|_*+\|g(x_t)\|_*)^2\leqslant2\|G(x_t,\xi_t)\|_*^2+2\|g(x_t)\|_*^2, \tag{5.52}$$

我们有

$$\mathbb{E}\left[\exp\left(\frac{\|G(x,\xi)\|^2}{M_*^2}\right)\right]\leqslant\exp\{1\}\quad\Rightarrow\quad\mathbb{E}\left[\exp\left(\frac{\|\Delta(x,\xi)\|^2}{(2M_*)^2}\right)\right]\leqslant\exp\{1\}. \tag{5.53}$$

利用 Markov 不等式的变形, 对任意 $t>0$, 有

$$\mathbf{Prob}\{X\geqslant\epsilon\}\leqslant\frac{\mathbb{E}[\mathrm{e}^{tX}]}{\mathrm{e}^{t\epsilon}}. \tag{5.54}$$

若 $\mathbb{E}[\mathrm{e}^X]\leqslant\exp(1)$, 对任意 $\lambda>0$, 有

$$\mathbf{Prob}\{X\geqslant1+\lambda\}\leqslant\frac{\mathbb{E}[\mathrm{e}^X]}{\exp(1+\lambda)}=\exp(-\lambda). \tag{5.55}$$

因此,

$$\mathbf{Prob}\{\|G(x,\xi)\|^2\geqslant(1+\lambda)M_*^2\}\leqslant\exp(-\lambda),\quad\mathbf{Prob}\{\|\Delta(x,\xi)\|^2\geqslant4(1+\lambda)M_*^2\}\leqslant\exp(-\lambda). \tag{5.56}$$

由收敛性估计不等式, 可得

$$\gamma_t[f(x_t)-f(x)]\leqslant V(x_t,x)-V(x_{t+1},x)+(2\nu)^{-1}\gamma_t^2\|G(x_t,\xi_t)\|^2+\gamma_t\langle\Delta_t,x_t-x\rangle. \tag{5.57}$$

将上式从 1 加到 k 得到

$$\begin{aligned}f(\bar{x}_1^k)-f^*&\leqslant\left(\sum_{t=1}^k\gamma_t\right)^{-1}\left[V(x_1,x^*)+\sum_{t=1}^k\frac{\gamma_t^2}{\nu}\|G(x_t,\xi_t)\|^2-\sum_{t=1}^k\gamma_t\langle\Delta_t,x_t-x^*\rangle\right]\\&\leqslant\left(\sum_{t=1}^k\gamma_t\right)^{-1}\left[V(x_1,x^*)+\sum_{t=1}^k\frac{\gamma_t^2}{\nu}\|G(x_t,\xi_t)\|^2+\sum_{t=1}^k\gamma_t\|\Delta_t\|\|x_t-x^*\|\right].\end{aligned} \tag{5.58}$$

取 $V(x,y)$、$D_{\omega,X}$ 和 $\{\gamma_t\}$ 满足如下:

$$V(x,y)\geqslant\frac{\nu}{2}\|x-y\|^2,\quad D_{\omega,X}^2=\sup_{x,y\in X}V(x,y),\quad\gamma_t=\frac{\sqrt{2\nu}D_{\omega,X}}{M_*k}. \tag{5.59}$$

在假设 $\mathbb{E}[\exp(\|G(x,\xi)\|_*^2/M_*^2)]\leqslant\exp(1)$ 下, 有

$$\mathbf{Prob}\left\{f(\bar{x}_1^k)-f(x_*)>\frac{\sqrt{2}M_*D_{\omega,X}(12+2\lambda)}{\sqrt{\nu N}}\right\}\leqslant 2\exp(-\lambda). \tag{5.60}$$

令上式右端等于 ϵ 解得 $\lambda(\epsilon)$ 并代入左边的估计值可以得到结论. □

引理 5.3 (Azuma-Hoeffding 不等式) 设 $Z_1,Z_2,\ldots$ 是一个鞅差序列 (bounded difference martingale sequence), 且对 $i=1,2,\ldots$ 满足 $|Z_i|\leqslant V$, 则下列不等式成立:

$$\mathbf{Prob}\left(\sum_{i=1}^N Z_i\geqslant c\right)\leqslant\exp\left(-\frac{2c^2}{NV^2}\right), \tag{5.61}$$

$$\mathbf{Prob}\left(\sum_{i=1}^N Z_i\leqslant c\right)\leqslant\exp\left(-\frac{2c^2}{NV^2}\right). \tag{5.62}$$

定理 5.7 若假设条件 (A1)–(A3) 成立, 则随机镜像下降法 (算法 5.5) 求得 (ϵ,δ) 近似解的复杂度为

$$\mathcal{O}\left(\frac{1}{\epsilon^2}\ln\frac{1}{\delta}\right). \tag{5.63}$$

证明 将不等式 (5.45) 左右两边从 $t=1$ 到 N 求和得

$$\gamma_t[f(x_t)-f(x)]\leqslant V(x_t,x)-V(x_{t+1},x)+\nu^{-1}\gamma_t^2\|G_t\|^2+\gamma_t\langle\Delta_t,x_t-x\rangle. \tag{5.64}$$

取 $\gamma_t\equiv\gamma$ 为常数, $V(x,y)$ 满足条件 $V(x,y)\geqslant\frac{\nu}{2}\|x-y\|^2$, 并令 $\bar{x}_N=\sum_{t=1}^N x_t/N$. 由于 $f(x)$ 的凸性以及假设条件 (A3), 我们可以推出

$$f(\bar{x}_N)-f^*\leqslant\frac{1}{N}\sum_{t=1}^N(f(x_t)-f^*)\leqslant\frac{1}{N\gamma}V(x_1,x^*)+\frac{\gamma M_*^2}{\nu}+\frac{1}{N}\sum_{t=1}^N\langle\Delta_t,x_t-x^*\rangle. \tag{5.65}$$

令 $D^2=\sup_{x,y\in X}V(x,y)$, 并代入上式, 可以得到

$$f(\bar{x}_N)-f^*\leqslant\frac{D^2}{N\gamma}+\frac{\gamma M_*^2}{\nu}+\frac{1}{N}\sum_{t=1}^N\langle\Delta_t,x_t-x^*\rangle. \tag{5.66}$$

令 $L=\frac{D^2}{N\gamma}+\frac{\gamma M_*^2}{\nu}$, 对任意常数 c, 得到

$$\mathbf{Prob}(f(\bar{x}_N)-f^*-L\geqslant c)\leqslant \mathbf{Prob}\left(\frac{1}{N}\sum_{t=1}^{N}\langle\Delta_t,x_t-x^*\rangle\geqslant c\right). \tag{5.67}$$

注意到 $\langle\Delta_t,x_t-x^*\rangle$ 是一个鞅差序列, 且 $\|\Delta_t\|_2=\|g_t-G_t\|_2\leqslant 2M_*$, 于是可以得到

$$|\langle\Delta_t,x_t-x^*\rangle|\leqslant\|\Delta_t\|_2\|x_t-x^*\|_2\leqslant 2M_*\sqrt{\frac{2V(x_t,x^*)}{\nu}}\leqslant\sqrt{\frac{8}{\nu}}M_*D. \tag{5.68}$$

由引理 5.3, 对任意常数 b, 得到

$$\mathbf{Prob}\left(\sum_{t=1}^{N}\langle\Delta_t,x_t-x^*\rangle\geqslant b\right)\leqslant\exp\left(-\frac{b^2\nu}{4NM_*^2D^2}\right). \tag{5.69}$$

取 $b=\sqrt{2N/\nu}M_*D\epsilon$, 可以得到

$$\mathbf{Prob}\left(\frac{1}{N}\sum_{t=1}^{N}\langle\Delta_t,x_t-x^*\rangle\geqslant\sqrt{\frac{2}{\nu N}}M_*D\epsilon\right)\leqslant\exp\left(-\frac{\epsilon^2}{2}\right). \tag{5.70}$$

令 $P=\sqrt{\frac{2}{\nu N}}M_*D\epsilon$, 由 (5.67) 得到

$$\mathbf{Prob}\left(f(\bar{x}_N)-f^*\geqslant L+P\right)\leqslant\exp\left(-\frac{\epsilon^2}{2}\right). \tag{5.71}$$

注意到当 $\gamma\equiv\frac{D}{M}\sqrt{\frac{\nu}{N}}$ 时, L 取极小值, 此时 $L=2M_*D/\sqrt{N\nu}$, 于是得到结论

$$\mathbf{Prob}\left\{f(\bar{x}_N)-f^*\geqslant\frac{2DM_*}{\sqrt{N\nu}}+\frac{2DM_*\sqrt{\ln\frac{1}{\delta}}}{\sqrt{N\nu}}\right\}\leqslant\delta. \tag{5.72}$$

利用上述不等式可以证明定理 5.7 的结论. □

我们将不同条件下随机次梯度算法的 (ϵ,δ) 计算复杂度总结到表 3.

5.4 光滑随机优化问题

5.4.1 凸随机优化问题

本小节仍旧考虑随机优化问题 (5.1), 但对目标函数 f 做如下假设:

(F1) $f(x)$ 光滑且是凸函数;

(F2) $f(x)$ 的梯度满足 $\|\nabla f(x)-\nabla f(y)\|_*\leqslant L\|x-y\|,\ \forall\ x,y\in X$.

并考虑 $f(x)$ 的随机梯度在如下假设条件下的收敛性.

表 3 随机次梯度法 (ϵ,δ) 计算复杂度的比较

假设条件	(ϵ,δ) 复杂度
$\|G(x,\xi_t)\|_*\leqslant M_*$	$\mathcal{O}(\frac{1}{\epsilon^2}\ln\frac{1}{\delta})$
$\mathbb{E}_{\xi_t}[\exp(\|G(x,\xi_t)\|_*^2/M_*^2)]\leqslant\exp(1)$	$\mathcal{O}(\frac{1}{\epsilon^2}\ln^2\frac{1}{\delta})$
$\mathbb{E}_{\xi_t}[\|G(x,\xi_t)\|_*^2]\leqslant M_*^2$	$\mathcal{O}(\frac{1}{\epsilon^2\delta^2})$

(A6) 存在子程序 $\mathcal{SFO}$ 对任意 $x\in X$ 和随机样本 $\xi\in\Xi$, 返回函数值 $F(x,\xi)$ 和随机梯度 $G(x,\xi)$. 且存在 $\sigma>0$, 对任意 $x\in X$, 有

$$\text{(i) } \mathbb{E}\left[G(x,\xi)\right]\equiv\nabla f(x), \tag{5.73}$$

$$\text{(ii) } \mathbb{E}\left[\|G(x,\xi)-\nabla f(x)\|_*^2\right]\leqslant\sigma^2. \tag{5.74}$$

(A7) 对任意 $x\in X$, 有

$$\mathbb{E}\left[\exp\left\{\frac{\|G(x,\xi_t)-\nabla f(x)\|_*^2}{\sigma^2}\right\}\right]\leqslant\exp(1), \tag{5.75}$$

其中条件 (A6) 说明随机梯度 $G(x,\xi_t)$ 是无偏的, 且方差小于 σ^2. 由 Jensen 不等式, 有如下包含关系:

$$\text{(A7)}\ \Rightarrow\ \text{(A6)(ii)}. \tag{5.76}$$

当 $f(x)$ 非光滑时, 我们利用条件 (A3), 即 $\|G(x,\xi_t)\|_*\leqslant M_*$, 得到

$$\begin{aligned}\mathbb{E}[\|G(x_t,\xi_t)\|_*^2]&=\|\mathbb{E}[G(x_t,\xi_t)]\|_*^2+\mathbb{E}[\|G(x_t,\xi_t)-\mathbb{E}[G(x_t,\xi_t)]\|_*^2]\\&=\|\nabla f(x_t)\|_*^2+\mathbb{E}[\|G(x_t,\xi_t)-\nabla f(x_t)\|_*^2]\\&\leqslant\|\nabla f(x_t)\|_*^2+\sigma^2=\|\nabla f(x_1)+\nabla f(x_t)-\nabla f(x_1)\|_*^2+\sigma^2\\&\leqslant 2\|\nabla f(x_1)\|_*^2+2\|\nabla f(x_t)-\nabla f(x_1)\|_*^2+\sigma^2\\&\leqslant 2\|\nabla f(x_1)\|_*^2+2L^2\|x_t-x_1\|^2+\sigma^2\\&\leqslant 2\|\nabla f(x_1)\|_*^2+4\nu^{-1}L^2D_{\omega,X}^2+\sigma^2.\end{aligned}$$

利用上述不等式, 我们可以得到 $\|G(x,\xi_t)\|_*$ 期望上界 M_* 的估计

$$\mathbb{E}\|G(x,\xi_t)\|_*\leqslant\mathcal{O}(1)\{\|\nabla f(x_1)\|_*+LD_{\omega,X}\nu^{1/2}+\sigma\}=M_*. \tag{5.77}$$

将上面关于 M_* 估计的等式代入 (5.47), 有

$$\mathbb{E}[f(\bar{x}_1^k)-f^*]\leqslant\mathcal{O}(1)\frac{D_{\omega,X}(\|\nabla f(x_1)\|_*+L\nu^{-1/2}D_{\omega,X}+\sigma)}{\sqrt{\nu k}}. \tag{5.78}$$

上述结论并没有利用 f 的光滑性. 事实上, 当 $\sigma=0$ 时, 随机优化问题变成确定性优化问题, 则上述收敛性结论和梯度法求解确定性光滑优化问题的收敛性不一致 (即满足 $\mathcal{O}(1/k)$ 的收敛速度). 通过利用 f 的光滑性、新的收敛性分析技巧和步长选择, 我们可以得到更好的收敛性结果. 考虑随机镜像梯度法输出如下解:

$$\tilde{x}_{k+1}=\frac{\sum_{t=1}^k\gamma_tx_{t+1}}{\sum_{t=1}^k\gamma_t}=\frac{\gamma_1x_2+\gamma_2x_3+\cdots+\gamma_kx_{k+1}}{\sum_{t=1}^k\gamma_t}. \tag{5.79}$$

我们可以证明随机梯度法在求解光滑随机凸优化问题时的复杂度. 首先引入一个辅助引理, 其证明可以在文献 [49] 中找到.

引理 5.4 设 $\xi_1,\ldots,\xi_2$ 是独立同分布的随机变量序列, 记 $\xi_{[t-1]}=(\xi_1,\ldots,\xi_t)$, 设 $Z_t=Z_t(\xi_{[t]})$ 是关于 $\xi_{[t]}$ 的 Borel 函数, 且 $\mathbb{E}[Z_t|\xi_{[t]}]=0$ 和 $\mathbb{E}[\exp\{Z_t^2/\sigma_t^2\}|\xi_{[t]}]\leqslant\exp(1)$ 几乎处处成立, 其中 $\sigma_t>0$ 为常数, 则对任意 $\lambda>0$, 有

$$\mathbf{Prob}\left(\frac{\sum_{t=1}^NZ_t}{\sum_{t=1}^N\sigma_t^2}\geqslant\lambda\right)\leqslant\exp\left\{-\frac{\lambda^2}{3}\right\}. \tag{5.80}$$

由上述引理, 我们可以证明如下复杂度结论 (参见文献 [16]).

定理 5.8 取步长 γ_t 满足 $0<\gamma_t<\nu/(2L)$, 令 $\tilde{x}$ 是随机镜像梯度法产生的解 (5.79), 则

(1) 在条件 (A6) 下,

$$f(\tilde{x}_{k+1})-f^*\leqslant K_0(k), \tag{5.81}$$

其中

$$K_0(k):=\frac{D_{\omega,X}^2+2\nu^{-1}\sigma^2\sum_{t=1}^k\gamma_t^2}{\sum_{t=1}^k\gamma_t}; \tag{5.82}$$

(2) 在条件 (A6) 和 (A7) 下,

$$\mathbf{Prob}\{f(\tilde{x}_{k+1})-f^*\geqslant K_0(k)+\lambda K_1(k)\}\leqslant \mathrm{e}^{-\lambda^2/3}+\mathrm{e}^{-\lambda}, \tag{5.83}$$

其中

$$K_1(k):=\frac{2D_{\omega,X}\sqrt{\frac{2\sigma^2}{\nu}\sum_{t=1}^k\gamma_t^2}+\frac{2\sigma^2}{\nu}\sum_{t=1}^k\gamma_t^2}{\sum_{t=1}^k\gamma_t}. \tag{5.84}$$

证明 令 $d_t=x_{t+1}-x_t$, $\Delta_t=G(x_t,\xi_t)-g(x_t)$, 注意到 $\nu\|d_t\|^2/2\leqslant V(x_t,x_{t+1})$,

$$\begin{aligned}
\gamma_t f(x_{t+1})&\leqslant\gamma_t\left[f(x_t)+\langle g(x_t),d_t\rangle+\frac{L}{2}\|d_t\|^2\right]\\
&=\gamma_t[f(x_t)+\langle g(x_t),d_t\rangle]+\frac{\nu}{2}\|d_t\|^2+\frac{L-\nu}{2}\|d_t\|^2\\
&\leqslant\gamma_t[f(x_t)+\langle g(x_t),d_t\rangle]+V(x_{t+1},x_t)+\frac{L-\nu}{2}\|d_t\|^2\\
&=\gamma_t[f(x_t)+\langle G(x_t,\xi_t),d_t\rangle]-\gamma_t\langle\Delta_t,d_t\rangle+V(x_{t+1},x_t)+\frac{L-\nu}{2}\|d_t\|^2\\
&\leqslant\gamma_t[f(x_t)+\langle G(x_t,\xi_t),d_t\rangle]+V(x_{t+1},x_t)+\frac{L-\nu}{2}\|d_t\|^2+\gamma_t\|\Delta_t\|_*\|d_t\|\\
&\leqslant\gamma_t[f(x_t)+\langle G(x_t,\xi_t),d_t\rangle]+V(x_{t+1},x_t)+\frac{L-\nu}{2}\|d_t\|^2+\frac{\gamma_t^2\|\Delta_t\|_*^2}{2(\nu-L\gamma_t)}.
\end{aligned} \tag{5.85}$$

由引理 5.4 有

$$\begin{aligned}
&\gamma_t[f(x_t)+\langle G(x_t,\xi_t),x_{t+1}-x_t\rangle]+V(x_{t+1},x_t)\\
&\quad\leqslant\gamma_t f(x_t)+[\gamma_t\langle G(x_t,\xi_t),x-x_t\rangle+V(x_t,x)-V(x_{t+1},x)]\\
&\quad=\gamma_t[f(x_t)+\langle g(x_t),x-x_t\rangle]+\gamma_t\langle\Delta_t,x-x_t\rangle+V(x_t,x)-V(x_{t+1},x)\\
&\quad\leqslant\gamma_t f(x)+\gamma_t\langle\Delta_t,x-x_t\rangle+V(x_t,x)-V(x_{t+1},x).
\end{aligned} \tag{5.86}$$

于是有

$$\gamma_t[f(x_{t+1})-f(x)]+V(x_{t+1},x)\leqslant\gamma_t\langle\Delta_t,x-x_t\rangle+V(x_t,x)+\frac{\gamma_t^2\|\Delta_t\|_*^2}{2(\nu-L\gamma_t)}. \tag{5.87}$$

将上式从 $t=1$ 到 k 求和并取 $x=x^*$,

$$\sum_{t=1}^k\gamma_t(f(x_{t+1})-f^*)\leqslant V(x_1,x^*)-V(x_{k+1},x^*)+\sum_{t=1}^k\left(\gamma_t\langle\Delta_t,x^*-x_t\rangle+\frac{\gamma_t^2\|\Delta_t\|_*^2}{2(\nu-L\gamma_t)}\right). \tag{5.88}$$

考虑到 f 的凸性, 有

$$f(\tilde{x}_{k+1}) = f\left(\frac{\sum_{t=1}^{k}\gamma_t x_{t+1}}{\sum_{t=1}^{k}\gamma_t}\right) \leqslant \frac{\sum_{t=1}^{k}\gamma_t f(x_{t+1})}{\sum_{t=1}^{k}\gamma_t}. \tag{5.89}$$

利用 $\gamma_t \leqslant \nu/(2L)$ 可以得到 $\nu/2 \leqslant 2(\nu - L\gamma_t)$, 将 (5.88) 两边同除 $\sum_{t=1}^{k}\gamma_t$ 并代入 (5.89), 得到

$$f(\tilde{x}_{k+1}) - f^* \leqslant \frac{V(x_1, x^*)}{\sum_{t=1}^{k}\gamma_t} + \frac{1}{\sum_{t=1}^{k}\gamma_t}\sum_{t=1}^{k}\left(\gamma_t\langle\Delta_t, x^* - x_t\rangle + \frac{2\gamma_t^2\|\Delta_t\|_*^2}{\nu}\right). \tag{5.90}$$

注意到 $(\tilde{x}_{k+1}, x_t)$ 是随机变量 $\xi_{[t-1]} = (\xi_1, \ldots, \xi_{t-1})$ 的函数, 因此有

$$\mathbb{E}[\langle\Delta_t, x^* - x_t\rangle \mid \xi_{[t-1]}] = 0. \tag{5.91}$$

由假设条件 (A7) 得 $\mathbb{E}[\Delta_t|\xi_{[t-1]}] \leqslant \sigma^2$, 对 (5.90) 两边关于随机变量 $\xi_{[t-1]}$ 求期望得

$$f(\tilde{x}_{k+1}) - f^* \leqslant \frac{D_{\omega,X}^2 + 2\nu^{-1}\sigma^2\sum_{t=1}^{k}\gamma_t^2}{\sum_{t=1}^{k}\gamma_t}. \tag{5.92}$$

证毕. □

有了上述定理, 我们可以通过选取特定的步长序列 $\{\gamma_t\}$ 得到收敛结果. 假设随机梯度算法的总迭代步数事先固定, 如 N 步. 我们取固定步长 $\gamma_t = \gamma$, 代入不等式 (5.81) 得

$$\mathbb{E}[f(x_{k+1}^{av}) - f^*] \leqslant \frac{D_{\omega,X}^2}{k\gamma} + \frac{2\gamma}{\nu}\sigma^2. \tag{5.93}$$

上式右端在约束 $\gamma \in (0, \nu/(2L)]$ 下关于 γ 求极小得

$$\gamma^* = \min\left\{\frac{\nu}{2L}, \sqrt{\frac{\nu D_{\omega,X}^2}{2k\sigma^2}}\right\}. \tag{5.94}$$

令 $\Omega_{\omega,X} = \sqrt{2/\nu}D_{\omega,X}$, 将 (5.94) 代入 (5.93) 得

$$\mathbb{E}[f(x_{k+1}^{av}) - f^*] \leqslant \frac{L\Omega_{\omega,X}^2}{k} + \frac{2\Omega_{\omega,X}\sigma}{\sqrt{k}}. \tag{5.95}$$

取 $\gamma_t = \gamma^*$, 代入不等式 (5.82) 和 (5.84) 得

$$K_0(k) = \frac{L\Omega_{\omega,X}^2}{k} + \frac{2\Omega_{\omega,X}\sigma}{\sqrt{k}}, \quad K_1(k) = \frac{2\Omega_{\omega,X}\sigma}{\sqrt{k}} + \frac{2\gamma\sigma^2}{\nu} \leqslant \frac{2\Omega_{\omega,X}\sigma}{\sqrt{k}} + \sqrt{\frac{2}{\nu}}D_{\omega,X}\frac{\sigma^2}{\sqrt{k\sigma^2}} = \frac{3\Omega_{\omega,X}\sigma}{\sqrt{k}}. \tag{5.96}$$

将上式代入 (5.83) 得

$$\mathbf{Prob}\left\{f(x_{k+1}^{av}) - f^* \geqslant \frac{L\Omega^2}{k} + \frac{\Omega\sigma}{\sqrt{k}}(2 + 3\lambda)\right\} \leqslant \mathrm{e}^{-\lambda^2/3} + \mathrm{e}^{-\lambda}. \tag{5.97}$$

取 λ 使得上式右端小于 $\delta > 0$, 我们可以得到光滑随机凸优化问题的 (ϵ, δ) 复杂度上界为

$$\mathcal{O}\left[\frac{L\Omega^2}{\epsilon} + \frac{\Omega^2\sigma^2}{\epsilon^2}\ln^2\frac{1}{\delta}\right]. \tag{5.98}$$

5.4.2 非凸随机优化问题

本小节介绍随机梯度法在求解非凸随机优化问题时的复杂度分析, 具体可参见文献 [15,17]. 为了方便分析, 我们将随机优化算法做一些变形, 见算法 5.6.

算法 5.6 随机迭代的随机梯度法 (randomized stochastic gradient (RSG) method)

输入: 初始点 $x_1 \in \mathbb{R}^n$, 最大迭代次数 N, 步长序列 $\{\gamma_k\}$, 随机迭代次数 R 的概率分布 $P_R(\cdot)$.

1. 设 R 是概率分布为 $P_R(\cdot)$ 的随机变量;
2. 对 $k = 1, \ldots, R$, 在 (x_k, ξ_k) 处调用子程序 $\mathcal{SFO}$ 得到随机梯度 $G(x_k, \xi_k)$, 并更新

$$x_{k+1} = x_k - \gamma_k G(x_k, \xi_k). \tag{5.99}$$

可以看到随机迭代的随机梯度法与随机梯度法的区别在于其总迭代步数事先确定, 且服从某一概率分布. 随机迭代的随机梯度法收敛性结论如下.

定理 5.9 选择步长 $\{\gamma_k\}$ 使得 $\gamma_k \leqslant 2/L$, 且概率密度函数 $P_R(\cdot)$ 取

$$P_R(k) := \mathbf{Prob}\{R = k\} = \frac{2\gamma_k - L\gamma_k^2}{\sum_{k=1}^N (2\gamma_k - L\gamma_k^2)}. \tag{5.100}$$

在条件 (A6) 下,

(1) 对任意的 $N \geqslant 1$, 有

$$\frac{1}{L}\mathbb{E}_{R,\xi_{[N]}}[\|\nabla f(x_R)\|^2] \leqslant \frac{D_f^2 + \sigma^2 \sum_{k=1}^N \gamma_k^2}{\sum_{k=1}^N (2\gamma_k - L\gamma_k^2)}; \tag{5.101}$$

(2) 若 f 是凸函数, 则对任意 $N \geqslant 1$, 有

$$\mathbb{E}_{R,\xi_{[N]}}[f(x_R) - f^*] \leqslant \frac{D_X^2 + \sigma^2 \sum_{k=1}^N \gamma_k^2}{\sum_{k=1}^N (2\gamma_k - L\gamma_k^2)}. \tag{5.102}$$

由上面的收敛性结论, 我们可以得到随机迭代的随机梯度法在求解非凸光滑的随机优化问题时的复杂度.

推论 5.2 对任意的常数 γ, 取步长 $\{\gamma_k\}$,

$$\gamma_k = \min\left\{\frac{1}{L}, \frac{\gamma}{\sigma\sqrt{N}}\right\}. \tag{5.103}$$

在条件 (A6) 下,

(1) 对任意的 $N \geqslant 1$, 有

$$\frac{1}{L}\mathbb{E}[\|\nabla f(x_R)\|^2] \leqslant C(N) := \frac{LD_f^2}{N} + \left(\gamma + \frac{D_f^2}{\gamma}\right)\frac{\sigma}{\sqrt{N}}; \tag{5.104}$$

(2) 若 f 是凸函数, 则对任意 $N \geqslant 1$, 有

$$\mathbb{E}[f(x_R) - f^*] \leqslant \frac{LD_X^2}{N} + \left(\gamma + \frac{D_X^2}{\gamma}\right)\frac{\sigma}{\sqrt{N}}. \tag{5.105}$$

利用 Markov 不等式 $\mathbb{E}[\|\nabla f(x_R)\|^2] \leqslant C(N)$, 随机迭代的随机梯度法运行一次的 (ϵ, δ) 复杂度为

$$\mathbf{Prob}\left\{\|\nabla f(x_R)\|^2 \geqslant \frac{L^2 D_f^2}{N}\lambda + \frac{2LD_f\sigma}{\sqrt{N}}\lambda\right\} \leqslant \frac{1}{\lambda}. \tag{5.106}$$

令 $\delta = 1/\lambda$, 并取

$$\frac{L^2 D_f^2}{N}\lambda + \frac{2LD_f\sigma}{\sqrt{N}}\lambda \leqslant \epsilon, \tag{5.107}$$

则随机迭代的随机梯度法 (ϵ, δ) 复杂度至多为

$$\mathcal{O}\left\{\frac{1}{\delta\epsilon} + \frac{\sigma^2}{\delta^2\epsilon^2}\right\}. \tag{5.108}$$

可以看到上述复杂度关于置信水平 δ 的结果不是很好, 但是我们可以提高它. 受到集成随机梯度法的启发, 我们也可以对随机迭代的随机梯度法进行集成, 即集成随机迭代的随机梯度法 (算法 5.7).

算法 5.7 集成随机迭代的随机梯度法 (two-phase RSG (2-RSG) method)

输入: 初始点 $x_1 \in \mathbb{R}^n$, 集成次数 S, 最大迭代次数 N, 采样次数 T.

对 $d = 1, \ldots, S$ 执行迭代,

1. 调用 RSG 算法, 其中输入为 x_1, 迭代最大次数为 N, 步长 $\{\gamma_k\}$, RSG 迭代次数 R 的概率分布为 $P_R(\cdot)$. 令 $\bar{x}_s$ 表示 RSG 的输出.

输出: 从 $\{\bar{x}_1, \ldots, \bar{x}_S\}$ 中选择 $\bar{x}^*$ 输出, 满足

$$\|g(\bar{x}^*)\| = \min_{s=1,\ldots,S}\|g(\bar{x}_s)\|, \quad g(\bar{x}_s) := \frac{1}{T}\sum_{k=1}^{T} G(\bar{x}_s, \xi_k). \tag{5.109}$$

定理 5.10 在条件 (A6) 下, 集成随机迭代的随机梯度法满足

(1) 令 $C(N)$ 按照推论 5.2 中定义, 对任意的 $\lambda > 0$, 有

$$\mathbf{Prob}\left\{\|\nabla f(\bar{x}^*)\|^2 \geqslant 2\left(4LC(N) + \frac{3\lambda\sigma^2}{T}\right)\right\} \leqslant \frac{S+1}{\lambda} + 2^{-S}; \tag{5.110}$$

(2) 令 $\epsilon > 0$, $\delta \in (0, 1)$ 给定, 若参数 (S, N, T) 取

$$S = S(\delta) := \left\lceil \log\left(\frac{2}{\delta}\right)\right\rceil, \tag{5.111}$$

$$N = N(\epsilon) := \left\lceil \max\left\{\frac{32L^2D_f^2}{\epsilon}, \left\lfloor 32L\left(\gamma + \frac{D_f^2}{\gamma}\right)\frac{\sigma}{\epsilon}\right\rfloor\right\}\right\rceil, \tag{5.112}$$

$$T = T(\epsilon, \delta) := \left\lceil \frac{24(S+1)\sigma^2}{\delta\epsilon}\right\rceil, \tag{5.113}$$

则集成随机迭代的随机梯度法为得到 (ϵ, δ) 近似解至多需要调用 $S(\delta)[N(\epsilon) + T(\epsilon, \delta)]$ 次 $\mathcal{SFO}$ 子程序.

综上, 随机梯度法在求解光滑函数时有如下 (ϵ, δ) 复杂度结论:

- 随机迭代的随机梯度法在条件 (A6) 下 (ϵ, δ) 复杂度为

$$\mathcal{O}\left\{\frac{1}{\delta\epsilon} + \frac{\sigma^2}{\delta^2\epsilon^2}\right\}. \tag{5.114}$$

- 集成随机迭代的随机梯度法在条件 (A6) 下 (ϵ, δ) 复杂度为

$$\mathcal{O}\left\{\frac{\log(1/\delta)}{\epsilon} + \frac{\sigma^2}{\epsilon^2}\log\frac{1}{\delta} + \frac{\log^2(1/\delta)\sigma^2}{\delta\epsilon}\right\}. \tag{5.115}$$

• 集成随机迭代的随机梯度法在条件 (A6) 和 (A7) 下 (ϵ, δ) 复杂度为

$$\mathcal{O}\left\{\frac{\log(1/\delta)}{\epsilon} + \frac{\sigma^2}{\epsilon^2}\log\frac{1}{\delta} + \frac{\log^2(1/\delta)\sigma^2}{\epsilon}\right\}. \tag{5.116}$$

6 结论

本文通过对优化算法复杂度分析的研究, 给出了不同优化算法在求解不同优化问题时效率的衡量准则, 并比较了不同算法的收敛性和复杂度. 第 1 节介绍了复杂度分析的理论基础, 给出了复杂度分析的概念、度量和具体例子, 构建了复杂度分析的理论框架. 后面的内容利用第 1 节的复杂度分析框架, 给出了优化问题的复杂度上界和复杂度下界. 第 2 节介绍光滑优化问题的复杂度分析, 给出了优化算法求解不同光滑函数集合时的收敛性和复杂度分析. 第 3 节介绍了非光滑优化问题的复杂度分析. 第 4 节介绍了条件梯度法的复杂度分析, 给出了优化问题在 $\mathcal{LO}$ 子程序下的复杂度下界, 以及加速的条件梯度法. 第 5 节介绍了随机优化算法的复杂度分析, 在第 1 节定义的随机优化问题解的度量下, 衡量了随机优化算法置信水平的复杂度.

本文对优化算法的收敛性和复杂度给出了全面的论述和介绍, 并将优化算法的收敛性分析纳入一个统一的框架里面, 即复杂度分析理论. 但本文还有一些领域没有考虑进来, 如复合函数 ($f(x) = g(x) + h(x)$, 其中 $g(x)$ 凸且光滑, $h(x)$ 凸但不一定光滑) 优化算法的收敛性和复杂度、鞍点问题的收敛性分析和二阶优化算法的收敛性分析. 我们将在以后的工作中继续考虑这些问题.

参考文献

1 Necoara I, Nesterov Y, Glineur F. Linear convergence of first order methods for non-strongly convex optimization. Math Program, 2019, 175: 69–107
2 Ben-Tal A, Nemirovski A. Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications, vol. 2. Philadelphia: SIAM, 2001
3 Nesterov Y. Introductory Lectures on Convex Optimization: A Basic Course, vol. 87. New York: Springer, 2013
4 Nemirovski A. Information-based complexity of convex programming. Lecture Notes, https://www2.isye.gatech.edu/~nemirovs/Lect_EMCO.pdf, 1995
5 Bubeck S. Convex optimization: Algorithms and complexity. FNT Mach Learn, 2015, 8: 231–357
6 Nemirovski A, Yudin D B. Problem Complexity and Method Efficiency in Optimization. New York: John Wiley & Sons, 1983
7 Lan G. Convex optimization under inexact first-order information. PhD Thesis. Atlanta: Georgia Institute of Technology, 2009
8 Frank M, Wolfe P. An algorithm for quadratic programming. Naval Res Logist, 1956, 3: 95–110
9 Lan G, Zhou Y. Conditional gradient sliding for convex optimization. SIAM J Optim, 2016, 26: 1379–1409
10 Lan G. The complexity of large-scale convex programming under a linear optimization oracle. arXiv:13095550, 2013
11 Nesterov Y. A method for unconstrained convex minimization problem with the rate of convergence $\mathcal{O}(1/k^2)$. Doklady AN USSR, 1983, 269: 543–547
12 Nesterov Y. Smooth minimization of non-smooth functions. Math Program, 2005, 103: 127–152
13 Tseng P. On accelerated proximal gradient methods for convex-concave optimization. Manuscript. Seattle: University of Washington, 2008
14 Beck A, Teboulle M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J Imag Sci, 2009, 2: 183–202
15 Ghadimi S, Lan G. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. Math Program, 2016, 156: 59–99
16 Lan G. An optimal method for stochastic composite optimization. Math Program, 2012, 133: 365–397

17 Ghadimi S, Lan G. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. SIAM J Optim, 2013, 23: 2341–2368
18 Hu C, Pan W, Kwok J T. Accelerated gradient methods for stochastic optimization and online learning. In: Proceedings of the 22nd International Conference on Neural Information Processing Systems. New York: Springer, 2009, 781–789
19 Levin A Y. An algorithm for minimizing convex functions. In: Doklady Akademii Nauk, vol. 160. Moscow: Russian Academy of Sciences, 1965, 1244–1247
20 Newman D J. Location of the maximum on unimodal surfaces. J ACM, 1965, 12: 395–398
21 Beck A, Teboulle M. Gradient-based algorithms with applications to signal recovery. In: Convex Optimization in Signal Processing and Communications. Cambridge: Cambridge University Press, 2009, 42–88
22 Nesterov Y. Gradient methods for minimizing composite functions. Math Program, 2013, 140: 125–161
23 Ghadimi S, Lan G, Zhang H. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. Math Program, 2016, 155: 267–305
24 Ahipaşaoğlu S D, Todd M J. A modified Frank-Wolfe algorithm for computing minimum-area enclosing ellipsoidal cylinders: Theory and algorithms. Comput Geom, 2013, 46: 494–519
25 Bach F, Lacoste-Julien S, Obozinski G. On the equivalence between herding and conditional gradient algorithms. arXiv:12034523, 2012
26 Beck A, Teboulle M. A conditional gradient method with linear rate of convergence for solving convex linear systems. Math Methods Oper Res, 2004, 59: 235–247
27 Clarkson K L. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. ACM Trans Algorithms, 2010, 6: 1–30
28 Freund R M, Grigas P. New analysis and results for the Frank-Wolfe method. Math Program, 2016, 155: 199–230
29 Jaggi M, Sulovsk M. A simple algorithm for nuclear norm regularized problems. In: Proceedings of the 27th International Conference on Machine Learning. Edinburgh: Omnipress, 2010, 471–478
30 Jaggi M. Sparse convex optimization methods for machine learning. PhD Thesis. Zürich: ETH Zürich, 2011
31 Jaggi M. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In: Proceedings of the 30th International Conference on Machine Learning. New York: ACM, 2013, 427–435
32 Linderoth J, Shapiro A, Wright S. The empirical behavior of sampling methods for stochastic programming. Ann Oper Res, 2006, 142: 215–241
33 Mak W K, Morton D P, Wood R K. Monte Carlo bounding techniques for determining solution quality in stochastic programs. Oper Res Lett, 1999, 24: 47–56
34 Shapiro A. Monte Carlo sampling approach to stochastic programming. In: ESAIM: Proceedings, vol. 13. Paris: EDP Sciences, 2003, 65–73
35 Shapiro A, Nemirovski A. On complexity of stochastic programming problems. In: Continuous Optimization. New York: Springer, 2005, 111–146
36 Verweij B, Ahmed S, Kleywegt A J, et al. The sample average approximation method applied to stochastic routing problems: A computational study. Comput Optim Appl, 2003, 24: 289–333
37 Robbins H, Monro S. A stochastic approximation method. Ann Math Statist, 1951, 22: 400–407
38 Ermoliev Y. Stochastic quasigradient methods and their application to system optimization. Stochastics, 1983, 9: 1–36
39 Polyak B T. New stochastic approximation type procedures. Automat Telemekh, 1990, 7: 2
40 Kushner H, Yin G G. Stochastic Approximation and Recursive Algorithms and Applications, vol. 35. New York: Springer, 2003
41 Polyak B T, Juditsky A B. Acceleration of stochastic approximation by averaging. SIAM J Control Optim, 1992, 30: 838–855
42 Ghadimi S, Lan G. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization, I: A generic algorithmic framework. SIAM J Optim, 2012, 22: 1469–1492
43 Ghadimi S, Lan G. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization, II: Shrinking procedures and optimal algorithms. SIAM J Optim, 2013, 23: 2061–2089
44 Nemirovski A, Juditsky A, Lan G, et al. Robust stochastic approximation approach to stochastic programming. SIAM J Optim, 2009, 19: 1574–1609
45 Johnson R, Zhang T. Accelerating stochastic gradient descent using predictive variance reduction. In: Advances in Neural Information Processing Systems. Washington: American Physiological Society, 2013, 315–323
46 Bottou L, Curtis F E, Nocedal J. Optimization methods for large-scale machine learning. SIAM Rev, 2018, 60: 223–311

47 Nesterov Y, Vial J P. Confidence level solutions for stochastic programming. Automatica, 2008, 44: 1559–1568
48 Nemirovski A, Shapiro A. Convex approximations of chance constrained programs. SIAM J Optim, 2007, 17: 969–996
49 Lan G, Nemirovski A, Shapiro A. Validation analysis of mirror descent stochastic approximation method. Math Program, 2012, 134: 425–458

Complexity analysis for optimization methods

Qichao Wang, Zaiwen Wen, Guanghui Lan & Ya-xiang Yuan

Abstract The subject on convergence analysis of optimization methods is an important area in optimization theory. However, convergence is not sufficient as a criterion for comparing the efficiency of different algorithms. Therefore, we need another theory to measure the difficulty of optimization problems and the efficiency of optimization algorithms. This theory is called complexity analysis theory of optimization algorithms. This paper is divided into five parts. The first section introduces the basic setups of complexity analysis frameworks, gives the definition, methods and examples of complexity analysis, and summarizes the complexity results. In the second section, the complexity analysis for the smooth optimization problem is introduced. We give the upper and lower complexity bounds for different optimization problems, and also introduce the framework of convergence analysis for the accelerated gradient method. In the third section, the upper bounds of the complexity of nonsmooth optimization problems are introduced. We present the complexity analysis for the subgradient method, the mirror descent method, the center-of-gravity method, and the ellipsoid method. In the fourth section, we introduce the complexity analysis for the conditional gradient method, study its upper and lower bounds of complexity and present the framework of the accelerated conditional gradient method. In the fifth section, we introduce the complexity analysis of the stochastic optimization algorithm, and compare the confidence level for the convergence under convex and nonconvex settings.

Keywords **optimization method, complexity analysis, accelerated gradient method, conditional gradient method, stochastic optimization method**

MSC(2010) **65K05, 65Y20, 90C15, 90C60**

doi: 10.1360/N012018-00251