

流形优化期末报告

张晨

2024 年 9 月 23 日

1 Question 1

2 Question 2

编程实现教材上的 Algorithm 3 和 Algorithm 4, 随机生成 Stiefel 流形的一个二次函数, 用 Algorithm 4 极小化这个二次函数, 比较两个算法的计算效果, 探索非单调的作用, 或者交替使用 BB 步长两个公式的作用.

2.1 随机二次函数的生成

首先考虑实现二次函数的随机生成问题. 对于整数 $n \geq p > 0$, 欧氏空间 $\mathbb{R}^{n \times p}$ 上的 *Stiefel* 流形为

$$\text{St}(n, p) = \{X \in \mathbb{R}^{n \times p} | X^T X = I_p\}.$$

对于矩阵 $B \in \mathbb{R}^{n \times p}$ 和对称正定矩阵 $A \in \mathbb{R}^{n \times n}$, $\text{St}(n, p)$ 上的二次函数定义为

$$f(X; A, B) = \text{tr}(X^T A X + X^T B).$$

故二次函数 f 的生成问题等价于矩阵 A, B 的生成问题. 在 MATLAB 中, 可以使用 **randn** 函数随机生成矩阵, 因此, 可以取

$$B \leftarrow \text{randn}(n, p).$$

但 MATLAB 中并没有直接生成对称正定矩阵的函数. 为了随机生成 A , 先考虑一个简单的引理:

引理 2.1 记集合 \mathcal{A} 为对称正定矩阵全体, 定义集合

$$\mathcal{B} = \{R^T R + D | R \in \mathbb{R}^{n \times n}, D = \text{diag}(d_1, \dots, d_n), 0 < d_1, \dots, d_n < 1\},$$

则 $\mathcal{A} = \mathcal{B}$.

证明: 首先证明 $\mathcal{B} \subseteq \mathcal{A}$. 对于任意的 $B = R^T R + D \in \mathcal{B}$, 有

$$B^T = (R^T R + D)^T = R^T R + D^T = R^T R + D = B,$$

故 B 是对称的. 对于任意的 $x \in \mathbb{R}_*^n$, 有

$$x^T Bx = (Rx)^T (Rx) + x^T Dx \geq x^T Dx \geq d_1 \|x\|_2^2 > 0.$$

因此, $B \in \mathcal{A}$.

再证明 $\mathcal{A} \subseteq \mathcal{B}$. 对于任意的 $A \in \mathcal{A}$, 记 $\lambda > 0$ 为 A 的最小特征值, 取 $d = \min(\lambda/2, 1/2)$, 则矩阵 $A - dI_n$ 仍为对称正定矩阵. 注意到, 根据对称正定矩阵的 Cholesky 分解, 存在矩阵 $R \in \mathbb{R}^{n \times n}$, 使得

$$A = (A - dI_n) + dI_n = R^T R + dI_n \in \mathcal{B}.$$

综上, $\mathcal{A} = \mathcal{B}$. ■

根据这个引理, 结合 MATLAB 中取值在 $(0, 1)$ 上的随机函数 **rand**, 可以得到对称正定矩阵的随机生成方法:

$$\begin{aligned} R &\leftarrow \text{randn}(n, n), \\ A &\leftarrow R^T R + \text{diag}(\text{rand}(n)). \end{aligned}$$

详细的代码实现可以参考附录. 在下文中, 我们考虑 $n = 500$, $p = 50$ 时各算法的表现, 取 MATLAB 中随机函数的种子为 99, 生成的矩阵对 (A, B) 同样可见附录.

2.2 梯度下降算法与 BB 算法的比较

本小节中, 我们使用基于单调线搜索的梯度下降算法与基于非单调线搜索的 BB 算法对二次函数 $f(X) = \text{tr}(X^T A X + X^T B)$ 进行优化. 在参数选择上, 我们使用表 1 中的参数值. 对于两个算法, 随机选取初始点 X_0 , 均使用基于 QR 算法的收缩映射.

表 1: 算法参数

参数	具体数值	参数描述
t_0	0.02	回退法的初始步长
ρ	0.5	回退幅度
c	0.001	Armijo 条件的系数
M	5	非单调线搜索的前项个数
α_{\max}	100	BB 步长的上界
α_{\min}	0.01	BB 步长的上界

我们先分析一下两个算法的收敛速度, 观察函数值 $f(X_k)$ 与点列 X_k 的误差变化情况, 结果见图 1, 其中左图为函数值 $f(X_k)$ 的相对误差随迭代次数的变化情况, 右图为点列 X_k 关于极小值点 X^* 的相对误差, 其度量为两点作为 $\mathbb{R}^{n \times p}$ 矩阵时, 差值 $X_k - X^*$ 的 Frobenius 范数. 从图 1 中可以看出, 两个算法都在 50 步内迭代到了极小值点, 精度达到了机器精度, 这两个算法的收敛速度都是线性收敛, 且 BB 算法的速度要显著快于梯度下降算法.

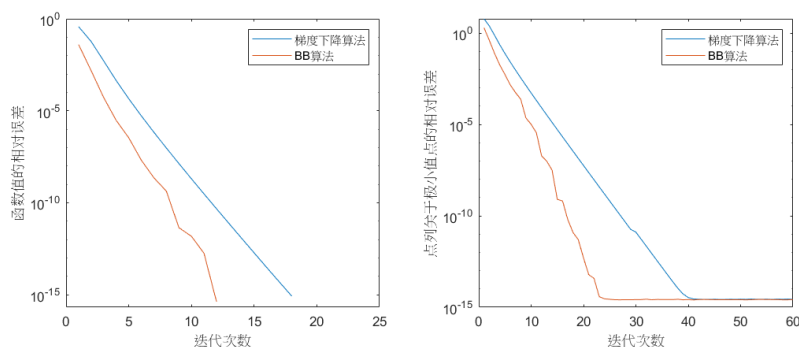
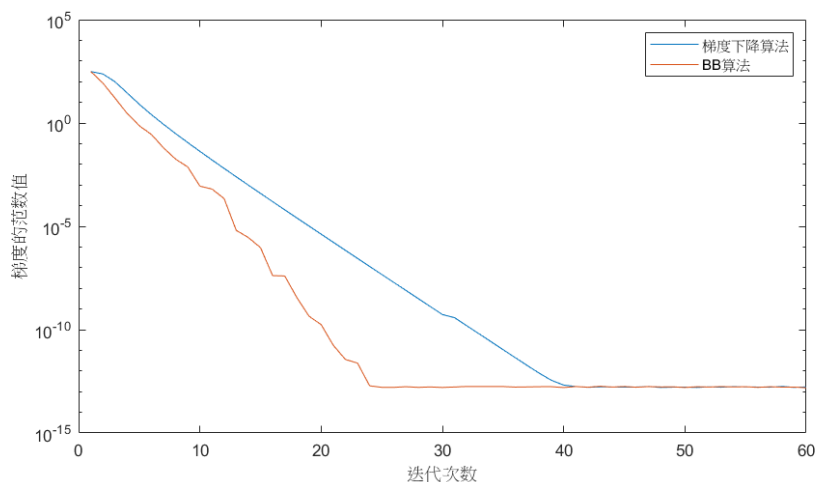


图 1: 相对误差的变化情况

值得注意的是, 在图 1 中, 函数值的收敛速度比点列的收敛速度更快, 这一点是由二次函数的性质产生的. 为了说明这一点, 我们来观察一下梯度值 $\text{grad}f(X_k)$ 随迭代次数的变化情况, 结果见图 2. 从图 2 中可以看到, 两个算法都在 50 步循环内使 Riemann 梯度 $\text{grad}f(X)$ 的范数达到了收敛值 10^{-13} , 且图像表现高度近似图 1 中点列的相对误差变化图. 为了解释这一现象, 考察 $\text{grad}f(X)$ 的表达式, 可以得到

图 2: $\text{grad}f(X)$ 的变化情况