数学实验 非线性优化 实验报告

计 65 赖金霖 2016011377

实验 7：

2.

说明：在 python 的 scipy.optimize.minimize 函数中，如果给定 Jacobi 矩阵，就会采用分析梯度，否则会采用数值梯度。

（1）

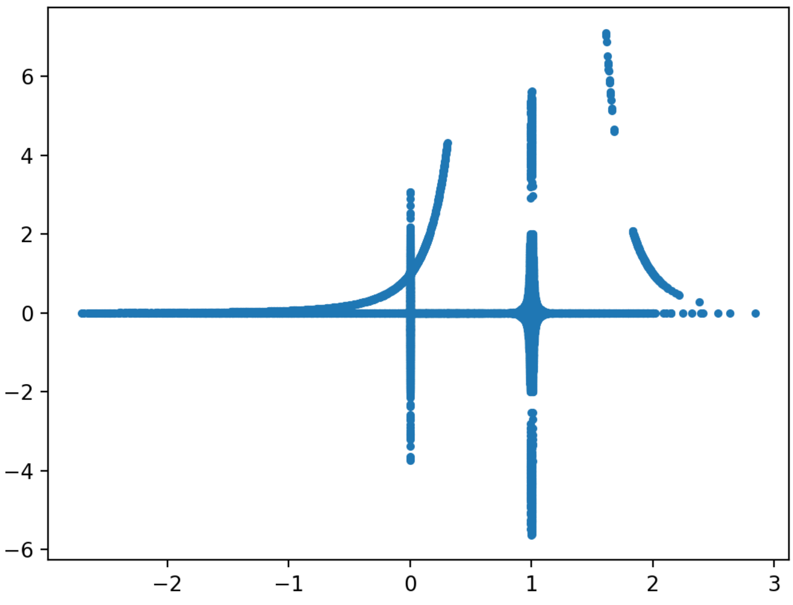
我们在[-2,2]×[-2,2]内均匀采 101×101 个点作为初值，进行实验。

使用不同的梯度和算法，有如下结果（结果为 min z=0）：

梯度	算法	平均迭代次数	时间
数值梯度	Nelder-Mead	65.62621	48.233s
	Powell	2.13322	20.554s
	CG	3.90628	15.566s
	BFGS	7.86884	14.216s
	Newton-CG		
分析梯度	Nelder-Mead		
	Powell		
	CG	5.03293	22.319s
	BFGS	7.26914	19.173s
	Newton-CG	10.85769	42.157s

可以看出，传统的 Nelder-Mead 算法花费了大量迭代次数，而后面的方法比如 Powell，共轭梯度和拟牛顿法在迭代次数和时间上都比较优秀，牛顿共轭梯度算法反而不如拟牛顿法。此外，不同的梯度策略也影响了算法的效率，整体上分析梯度的耗时比数值梯度长（可能是因为函数比较复杂），迭代次数却不一定更优秀。因此在随后的实验中我采用 BFGS 算法进行计算。

BFGS 算法在此题中得到的局部极小点分布如下（横轴为 x1，纵轴为 x2）：



可以验证，局部极小值位于 x1=0，x1=1，x2=0，$1-x1-x2(1-x1)^5$这几条曲线上。

（4）

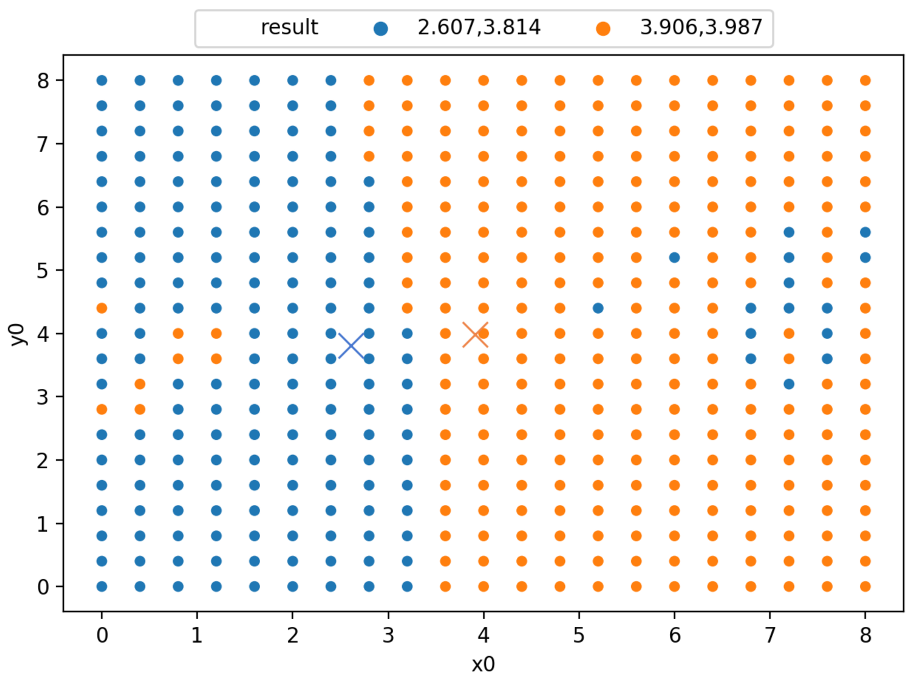
我们在[0,8]×[0,8]内采 21×21 个点，进行实验，发现此函数有两个局部极小点(2.607,3.814)和(3.906,3.987)，其函数值分别为-1.72218 和-1.77515。所以此函数的全局最小点在(3.906,3.987)，全局最小值为-1.77515。

不同算法和不同梯度的效率如下：

梯度	算法	平均迭代次数	时间
数值梯度	Nelder-Mead	46.87302	3.343s
	Powell	3.13832	2.533s
	CG	6.53288	2.777s
	BFGS	7.63946	2.444s
	Newton-CG		
分析梯度	Nelder-Mead		
	Powell		
	CG	6.53288	2.252s
	BFGS	7.63946	2.101s
	Newton-CG	5.68934	2.426s

可以看出，对本题的函数，梯度策略不会改变迭代次数，这可能是因为函数形式比较简单，数值梯度拟合较好。此外，我们也能注意到，对比较简单的函数，分析梯度效率比数值梯度高。

采用 BFGS 算法，可以绘制采样点最终迭代到的极小值点（×为两个极小值点，颜色表示迭代结果），如下：



可以看出，大多数点迭代到和它最近的极小值点，但有少数例外，可能是因为梯度较大使得迭代过程越过最近的极小值点。

实验 9：

3.

本题一律采用分析梯度和 BFGS 算法计算结果。

初值	条件	x1	x2	x3	x4	极小值
(-3,-1,-3,-1)	(1)	1	1	1	1	0
	(2)	-1.10823	1.23717	0.88038	0.77431	4.48982
	(3)	1.77475	-1.77475	1.42763	-3.55817	5782.5759
(3,1,3,1)	(1)	1	1	1	1	0
	(2)	1.42004	-0.19036	1.46609	2.15117	487.98692
	(3)	-1.39818	1.39818	-2.03504	4.0218	164.89552
(2,2,2,2)	(1)	1	1	1	1	0
	(2)	1.42004	-0.19036	1.46609	2.15117	487.98692
	(3)	1.31762	0.15682	3.13938	9.57399	867.75885

通过这些对比试验，我们可以得到如下结论：

1. 在没有约束条件下，一个函数的最小值和极小值是固定的。增加约束条件后，可能会生成新的极小值，最小值大小也可能会上升。增加不同的约束条件，生成的极小值一般不同。
2. 对同一个约束条件，不同的初值迭代得到的极小值不一定相同。
3. 对同一个初值，在不同约束条件下迭代得到的极小值一般不同。

8.

假设 A 股票的年收益率为随机变量 A，B 股票的年收益率为随机变量 B，C 股票的年收益率为随机变量 C，通过往年数据，可以计算得出：

D(A)=0.01080754

D(B)=0.0583917

D(C)=0.09422681

Cov(A,B)= 0.01240721

Cov(B,C)= 0.05542639

Cov(C,A)= 0.01307513

E(A)= 0.08908333333333333

E(B)= 0.21366666666666667

E(C)= 0.23458333333333334

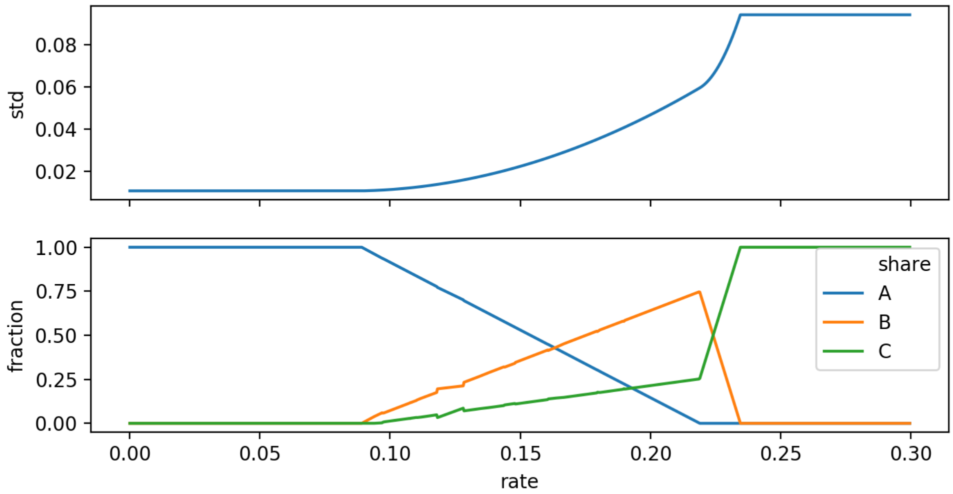
假设 A、B、C 股票的投资占比分别为 x1,x2,x3，如果期望年收益率至少为 p，那么我们的最优投资组合需要满足在 x1E(A)+x2E(B)+x3E(C)≥p 且 x1+x2+x3=1 的情况下，最小化 D(x1A+x2B+x3C)。

当 p=0.15 时，可以计算得出最优投资组合为 x1=0.53009261，x2=0.35640757，x3=0.11349982，D(x1A+x2B+x3C)= 0.022413776846355128。

（1）

容易看出，当年收益率大于 E(C)时，投资组合已经没有意义，而只有收益率小于 E(A)时，投资组合才会没有意义，所以我们可以在 0%~30%内均匀采 1001 个

点，查看投资组合的变化：



容易看出，在某一个节点上，平均收益率少于期望收益率的股票比例下降，平均收益率高于期望收益率的股票比例上升。而风险始终随期望收益率的上升而上升。

(2)

设国库券的购买比例为 x4，我们只需要把条件改为 x1E(A)+x2E(B)+x3E(C)+0.05x4≥p 且 x1+x2+x3+x4=1，此时最优投资策略为 x1=0.08455247,x2=0.42936896,x3=0.14314413,,x4=0.34293444，风险（方差）为 0.020803518257584055。

(3)

对于某个期望的投资组合 x,其交易费用为 sum(abs(x-[0.5,0.35,0.15]))/2,故我们只需把条件改为 x1E(A)+x2E(B)+x3E(C)≥p+ sum(abs([x1,x2,x3]-[0.5,0.35,0.15]))/200 且 x1+x2+x3+ sum(abs([x1,x2,x3]-[0.5,0.35,0.15]))/200=1（手续费需要扣除）。此时最优投资策略为 x1= 0.526475883,,x2= 0.349998299,x3=0.122990986，风险（方差）为 0.02261142308530563。

附录

代码可以在 https://github.com/1116924/math_exp/blob/master/exp6/nonlinear_optimization.py 找到。