Project 1单纯形法的实现

运筹学课程作业

数学规划求解器：国内外现状

- 1939年，苏联数学家和经济学家Leonid Kantorovich发明线性规划
- 1979年芝加哥大学Charge 发布Lingo
- 1983年英国爱丁堡大学Ashford创建XPRESS
- 1987年美国莱斯大学Bixby创建CPLEX公司
- 2000年COIN-OR成立，开放源代码CLP和CBC
- 2005年德国ZIB发布了开源整数规划工具SCIP
- 2008年Cplex创始人Bixby离开创办GUROBI
- 2017年，上海交大创建一个开源数学规划工具LEAVES
- 2018年中科院推出CMIP

美国与英国三大求解器巨头，累计三十年研发历史和95%以上市场。线性、整数、非线性模块功能齐全

- **GUROBI**
- **CPLEX**
- **XPRESS**
- SAS： 最大的商业统计软件（北卡）
- CVX：最著名的求解器建模平台（斯坦福）
- IPOPT：著名的非线性规划开源求解全局（卡耐基梅隆）
- Coin-OR：最好的开源线性规划（多组织维护）
- Baron： 最好的非线性规划（多组织维护）
- NEOS：最大的优化求解免费平台
- SCIP：最好的整数规划
- CBC：美国
- SOPLEX：德国
- MOSEK：丹麦
- GLPK： 俄罗斯

数学规划问题

线性规划

- 规模较大：数百万变量和约束
- 求解算法：单纯形法、内点法
- 稀疏性质：大量稀疏矩阵
- 受到精度影响
- 占大概15%

整数规划

- 是经典的NP-完全问题，难度最大
- 复杂多样的问题结构
- 求解算法：分支定界、割平面方法等
- 大概79%

非线性规划

- （混合整数）二阶锥规划
- （混合整数）二次规划
- （混合整数）半正定规划
- 大概6%

线性规划算法

单纯形法

- 1940年代推出
- 基于LU分解 ，求解非对称线性方程
- 每个循环较快，但是循环多
- 适合用于稀疏问题

内点法

- 1980年代提出
- 基于Cholesky分解
- 每个循环慢，但是只需要数十至数百循环
- 适用于较难的问题、非稀疏、高度退化问题

Matlab 的线性规划求解函数

Linprog(f, A, b, Aeq, beq, lb, ub)

$$\min_{x} f^T x \text{ such that } \begin{cases} A \cdot x \le b, \\ Aeq \cdot x = beq, \\ lb \le x \le ub. \end{cases}$$

f、x、b、beq、lb 和 ub 是向量，A 和 Aeq 是矩阵。

For this example, use these linear inequality constraints:

$$x(1) + x(2) \le 2$$
$$x(1) + x(2)/4 \le 1$$
$$x(1) - x(2) \le 2$$
$$-x(1)/4 - x(2) \le 1$$
$$-x(1) - x(2) \le -1$$
$$-x(1) + x(2) \le 2.$$

```
A = [1 1
    1 1/4
    1 -1
    -1/4 -1
    -1 -1
    -1 1];

b = [2 1 2 1 -1 2];
```

Project1 线性规划要求

总分10分 语言要求：python或者c++

- 代码规范性2分（包括**命名规范和函数模块化**）+正确性4分：
  - **模块0**：转换标准形式
  - **模块1**： 检查A是否行慢秩，如果非满秩，移除多余的约束
  - **模块2**： 初始化可行基解：大M法
  - **模块3**：单纯形法的迭代
  - 正确性还包括避免退化解循环，需要验证并写入报告。
- 实验报告4分
  - **说明**各个模块中，是如何处理的，可以结合讲义。
  - 随机生成**测试案例**，报告求解时间随着问题规模的变化图像。每个规模的时间，需要对**随机的20个测试案例取均值**，以及计算标准差。遇到无可行解问题直接跳过，不统计求解时间。
  - 要求图片**清晰**。

简单的前置处理

- A是否有全为0的行，此时b如何？
- 是否有简单的线性相关行 $[a_i\ b_i] = \lambda[a_j\ b_j]$
- A是否有全为0的列

模块1：检查A是否满秩：方法1

给定A$x \le b$，如何得到所有行都是线性无关的矩阵？

可以考虑这个问题：

$$
\begin{array}{rll}
f^* = & \text{maximize} & s^T x \\
& \text{subject to} & Ax \le b \\
& & s^T x \le t + 1.
\end{array}
$$

不等式$s^T x \le t$ 相对于 $Ax \le b$是冗余的，即s与A线性相关，当且仅当f*小于等于t

该结论的提示：可以先从小的线性系统出发，例如$A_0 x \le b_0$，这里A_0的行数很小，逐个增加新的不等式. 直到A的所有都行检查过。

模块1：检查A是否满秩：方法2

使用QR分解

- $\mathrm{A} \in \mathbf{R}^{\{\mathrm{m} \times \mathrm{n}\}}$，记$\mathrm{B} = A^T$
- 有如下分解：$\mathrm{B} = \mathrm{QR}$，其中$\mathrm{Q} \in \mathbf{R}^{\{\mathrm{n} \times m\}}$ 是正交矩阵，R的对角元的绝对值从大到小排列。
- 如果 R的对角元都非零，则A满秩。
- 反之，R有对角元为零，说明可以删掉对应的Q的列,得到B’。
- 最后得到满秩矩阵 $\mathrm{A}' = (B')^T$
- 向量b删掉对应的行元素即可

模块1：检查A是否满秩：方法？

- 你有何种方法？

可以通过检索文献，直接实现，说明大概是如何处理的即可。无需比前两种更优越。

通过如下代码检查你的函数：

- % Define a matrix A with linearly dependent rows
- A = [1 2 3 4; 1 1 3 4; 2 3 6 8];
-
- % Define a corresponding vector b
- b = [6; 12; 15];

下一页继续。。。

模块1：检查A是否满秩：方法？

- % Display the original matrix and vector
- disp('Original Matrix A:');
- disp(A);
- disp('Original Vector b:');
- disp(b);

- % Apply the makeFullRankQR function
- [A_fullrank, b_fullrank] = makeFullRankQR(A, b);
- % Display the modified matrix and vector
- disp('Modified Full Rank Matrix A:');
- disp(A_fullrank);
- disp('Modified Vector b:');
- disp(b_fullrank);
-

模块2：初始可行基解

大M法

- 如何确定M的大小？
  - 太小：非等价于原始问题
  - 太大：容易出现数值误差
  - 动态更新？
- 是否会添加冗余的列？

或许两阶段法更合适？

模块3： 单纯形法的迭代

- 给定可行基解 x，对应基坐标集合B和非基坐标N
- 计算最优判定条件
  - $r_{cost} = c_j - c_B^T B^{-1} A_j,\ j \in N$. 确定入基变量
  - 计算$-\frac{x_{B_i}}{d_{B_i}}$确定出基变量
  - 避免求逆矩阵
- 采用Bland’s rule避免陷入循环

回顾标准形式的分类

可行域为空 $\Longrightarrow$ 无解

可行域有界 $\Longrightarrow$ 唯一解 或者 不唯一解

可行域无界 $\Longrightarrow$ (1) 唯一解 或者 (2) 无穷多解,P 中可能不存在极点 或者 (3) 最优值为 $-\infty$

Theorem 3.4 (可行基解最优性) 考虑线性规划问题，在多面集 $P=\{x \mid Ax \geq b\}$ 上，最小化 $c^\top x$。假设 P 中存在至少一个极点。那么，要么最优值是 $-\infty$，要么必定有某个极点是最优解。

需包含以下测试案例

- 1.有可行解。例如：

```
A = [1 2 2 1 0 0;
     2 1 2 0 1 0;
     2 2 1 0 0 1];
b = [20; 20; 20];
c = [-10; -12; -12; 0; 0; 0];
```

```
Solution x:
     4
     4
     4
     0
     0
     0

Objective function value:
  -136

Exit flag (1 = solution found; -1 = unbounded; 0 = no feasible solution):
     1
```

需包含以下测试案例

- 2. 有冗余秩

```
A = [1 2 3;
     2 4 6;
     1 1 1];
b = [6; 12; 3];
c = [1; 2; 3];
```

```
Solution x:
     0
     3
     0

Objective function value:
     6

Exit flag (1 = solution found; -1 = unbounded; 0 = no feasible solution)
     1
```

需包含以下测试案例

- 无解：无可行域

```
A = [1 0 0;
     0 1 0
     0 0 1
     1 0 1];
b = [2; 2; 2; 2];
c = [1; 1; 1];
```

```
LP has no feasible point!
```

- 无解：无界

```
A = [1 -1 ;
     -1 1 ];
b = [0; 0;];
c = [-1; 0];
```

```
unbounded LP
```

Bland’s rule

- 一直选取满足出基入基条件最小的下标
- 测试案列：讲义中的循环例子，与采用Bland’s rule的测试结果。

Example 4.3 假设我们有如下初始单纯形表：

		x_1	x_2	x_3	x_4	x_5	x_6	x_7
	3	$-3/4$	20	$-1/2$	6	0	0	0
$x_5 =$	0	$1/4^*$	-8	-1	9	1	0	0
$x_6 =$	0	$1/2$	-12	$-1/2$	3	0	1	0
$x_7 =$	1	0	0	1	0	0	0	1

采用如下规则选取出基入基下标：

1. 对于非基向量，选取下标 j 使得减少量 $\bar{c}_j$ 最小
2. 对于基向量，出基下标，选取所有满足条件中下标最小的。

如何处理存在退化可行基解情况 若遇到退化基解，可采取 Bland’s rule[最小下标转轴规则] 避免循环:

1. 若非基向量中，存在多个 $\bar{c}_i < 0, i \in N$. 选取 j 为最小的下标。
2. 出基过程中，即在计算下标 r 使得 $\theta^* = \min_{\{i|d_{B_i}<0\}}(-\frac{x_{B_i}}{d_{B_i}})$ 时，选取最小的下标 B_r.