Lecture 9

§1 两阶段法的单纯形表实现

1. Phase I 的处理

进行两阶段法时，我们先要解下面的优化问题：

$$\begin{aligned} \text{minimize}_{x,y} \quad & e^T y \\ \text{s.t.} \quad & Ax + y = b \\ & x, y \geq 0 \end{aligned}$$

① Step 1：调整原问题的 constraint $Ax = b$，使 b 中元素均非负

② Step 2：加入 y，构建 auxiliary problem

③ Step 3：对 auxiliary problem 构建 initial 单纯形表

- Bottom part：即为 $Ax + y = b$ 的系数
- Reduced cost part：（注意这里 y 不为 slack variable，reduced cost 不为目标函数系数）
  - 对于 basic part：reduced cost 为 0
  - 对于 nonbasic part：有 $\bar{c}_j = c_j - c_B A_B^{-1} A_j = -e^T A_j$，因此 reduced cost 即为该列元素和的相反数
- Objective value part：即为该列元素和的相反数

④ Step 4：解单纯形表

2. Phase II 的处理

若经过 Phase I，得到 optimal solution $(x^*, 0)$，optimal value 0.

① 若 x^* not degenerate：

Step 1：直接去除关于 auxiliary variables 的列.

Step 2：利用 $\bar{c}_j = c_j - c_B^T A_B^{-1} A_j$ 重新求解 top row

Step 3：重新计算 objective value（注意表右上角为 objective value 相反数）

Step 4：解单纯形表

② 若 x^* degenerate：

将与 auxiliary variables 对应的 basic indices 替换为任意 orignal variables 的 indices，并重新计算表.

例：用单纯形表解下列优化问题

$$\begin{aligned} \text{minimize} \quad & x_1 + x_2 + x_3 \\ \text{subject to} \quad & x_1 + 2x_2 + 3x_3 = 3 \\ & -4x_2 - 9x_3 = -5 \\ & 3x_3 + x_4 = 1 \\ & x_1, \ x_2, \ x_3, \ x_4 \geq 0 \end{aligned}$$

先将 b 化为正，并引出 auxiliary problem：

$$
\begin{aligned}
\text{minimize} \quad & x_5 + x_6 + x_7 \\
\text{subject to} \quad & x_1 + 2x_2 + 3x_3 + x_5 = 3 \\
& 4x_2 + 9x_3 + x_6 = 5 \\
& 3x_3 + x_4 + x_7 = 1 \\
& x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0
\end{aligned}
$$

列出 initial tableau 并解单纯形表：

B	-1	-6	-15	-1	0	0	0	-9
5	1	2	3	0	1	0	0	3
6	0	4	9	0	0	1	0	5
7	0	0	3	1	0	0	1	1

$\Rightarrow$

B	0	-4	-12	-1	1	0	0	-6
1	1	2	3	0	1	0	0	3
6	0	4	9	0	0	1	0	5
7	0	0	3	1	0	0	1	1

$\Rightarrow$

B	0	0	-3	-1	1	1	0	-1
1	1	0	-3/2	0	1	-1/2	0	1/2
2	0	1	9/4	0	0	1/4	0	5/4
7	0	0	3	1	0	0	1	1

$\Rightarrow$

B	0	0	0	0	1	1	1	0
1	1	0	0	1/2	1	-1/2	1/2	1
2	0	1	0	-3/4	0	1/4	-3/4	1/2
3	0	0	1	1/3	0	0	1/3	1/3

因此，$X = (1, 1/2, 1/3, 0)$ is a BFS for the original problem ($B = \{1, 2, 3\}$)

列出原问题的 tableau 并解单纯形表：

B	0	0	0	-1/2	-11/6
1	1	0	0	1/2	1
2	0	1	0	-3/4	1/2
3	0	0	1	1/3	1/3

$0 - (1,1,1) \cdot \begin{bmatrix} 1/2 \\ -3/4 \\ 1/3 \end{bmatrix}$

$\Rightarrow$

B	0	0	1/4	0	-7/4
1	1	0	-3/2	0	1/2
2	0	1	9/4	0	5/4
4	0	0	3	1	1

Thus, the optimal solution is $(1/2, 5/4, 0, 1)$ with optimal value $-7/4$

§2 单纯形法的复杂度

1. Complexity of an algorithm

- 程序解决问题所需的 number of arithmetic operations（加、减、乘、除、比较…）被称为 complexity of the algorithm
- 通常考虑 worst-case instance

e.g. Examples:

- Find the largest element among n numbers has complexity n
- Add two m-by-n matrices has complexity mn
- Multiply two n-by-n matrices in a naive way has complexity n^3

2. Polynomial-time algorithms

- 一个程序被称为 polynomial-time algorithm 若其解决问题所需的 number of arithmetic operations 的上界为一个关于 input size 的多项式
- polynomial-time algorithm 被认为 practical，否则被认为 impractical

e.g. Here are some known polynomial algorithms:

- Gaussian elimination for matrix inversion (n^3)
- Fast method for matrix inversion ($\sim n^{2.373}$)
- Naive method for sorting n numbers (n^2)
- Merge sort for sorting n numbers ($n \log n$)

Here are some non-polynomial algorithms:

- Enumeration method for traveling salesman problem ($n!$)
- Dynamic programming for traveling salesman problem ($2^n n^2$)

3. Definition: polynomial-time solvable problem
   - 若一个问题存在 polynomial-time algorithm，则我们称其为一个 polynomial-time solvable problem，或表示为 P
   - 存在一类问题，目前无法找到 polynomial-time algorithm，这类问题被称为 NP-Hard problem.
     若可以证明一类 NP-Hard problems 中的一个有 polynomial-time algorithm，则可证明这一类问题（NP-complete）都存在 polynomial-time algorithm （NP=P）
4. 单纯形法的复杂度
   - Pivoting rule 的选取会影响单纯形法的复杂度
   - 但截至目前，还没有发现 polynomial-time algorithm
   - 但在现实中表现很好，平均情况下 stops in a polynomial number of iterations
5. 线性规划的复杂度
   - 第一个 polynomial-time algorithm for LP: ellipsoid method
     但在现实中表现很差
   - 一个理论上与现实中均表现很好的 algorithm: interior point method