simplex法の基礎

January 20, 2019

1　基礎事項

この section では，線形計画問題に関する基礎事項を述べる．入りとして，数理計画問題の分類からスタートし，simplex 法の立ち位置について述べる．そののちに，以降の準備として，線形計画問題の基本形や基礎用語について述べる．

1.1　simplex 法の立ち位置

一般に数理計画問題は以下のように表すことができる．

$$\begin{aligned} min_{x,y} \quad & f(x,y) \\ s.t. \quad & g_i(x,y) \leq 0 \ (i \in \{1, \cdots, m\}) \\ & x \in \mathbb{R}^n, y \in \mathbb{Z}^\ell \ . \end{aligned} \tag{1}$$

日本語に直すと，「コスト $f(x,y)$ を最小にする連続変数 x と整数変数 y の組み合わせを見つけたい．ただし，m 個の条件 $g_k(x,y) \leq 0$ を全て満足する x, y でなければならない」となる．言葉だけの問題だが，コストのことを数理計画では目的関数 (objective) と呼び，条件のことを制約 (costraint) と呼ぶので，今後はこの言葉を使っていくものとする．

さて，(1) 式が数理計画問題の最も一般的な問題であるが，まずはこれを分類する．分類の仕方は簡単で，以下の三つの観点で分類される．

- 整数変数があるかどうか．同じだが $\ell = 0$ or $\ell \neq 0$ か．
- 目的関数 $f(x,y)$ が linear か quadoratic か nonlinear か．
- 制約式 $g_k(x,y)$ が linear か nonlinear か．

具体的に分類すると以下のようになり，それぞれ以下のような名前が付いている．

- 整数変数がない ($\ell = 0$).
  - 目的関数と制約式全てが linear → Linear Programming(LP)
  - 目的関数が quadratic で制約式全てが linear → Quadoratic Programming(QP)
  - それ以外 → Non-linear Programming(NLP)
- 整数変数がある ($\ell \neq 0$).
  - 目的関数と制約式全てが linear → Mixed Integer Linear Programming(MILP)
  - 目的関数が quadratic で制約式全てが linear → Mixed Integer quadoratic Programming(MIQP)
  - それ以外 → Mixed Intger Non-linear Programming(MINLP)

このうちで，LP, QP, NLP, MILP, MIQP については一般的なアルゴリズムが知られている．つまり，数式に落とすことさえできれば，とりあえず解くこと自体は可能である[1]．この中で，LP についてはいくつか効率的なアルゴリズムが知られているが，そのうちの一つが simplex 法であり，今回紹介するアルゴリズムである．

1.2 線形計画問題の基本形

先に述べたように，simplex 法は LP を解くアルゴリズムであるが，特に以下の形の問題を解くアルゴリズムである．

$$\begin{aligned} \min_x \quad & c^T x \\ s.t. \quad & \sum_j A_{ij} x_j = b_i \ (i \in \{1, \cdots, m\}), \\ & x \geq 0, x \in \mathbb{R}^n \end{aligned} \tag{2}$$

勿論，$c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ は constant な parameter で，解きたい問題に応じて与えるものである．なお，この形で書く場合，通常 $m < n$ を仮定することが常である．例えば $m = n$ かつ A が full rank であれば，(2) 式はもうすでに解けているので[2]．

[1] 勿論，十分高速に解けるかどうかは微妙である．

[2] 勿論，これはこの形で書いた場合．不等式制約が残っている場合には $m < n$ を満たす必然性は存在しない

このような書き方をすると,「おいおい，これは一般的な形なのかよ？」と感じる人も多いかと思うが，実はこれで一般的な形である．以下ではそのことを確認する．

まず，今回は LP を考えている，つまり，objective も constraint も線形なので，大雑把には (2) 式のようになるのは想像できるかと思う．ということで気になるのは以下の三点かと思われる．

1. objective 最大化はできないの?
2. $x \geq 0$ は限定しすぎでは？
3. 等号制約しか考えれないの?

まずは，1 から．これについては簡単で，$f(x)$ 最大化は $-f(x)$ の最小化と思えば良いので，最小化問題に帰着させることができる．

続いて 2 について．これも比較的簡単で $x \in \mathbb{R}^n$ については，二つの postive な変数 x', x'' $(x' \geq 0, x'' \geq 0)$ を使って $x = x' - x''$ と表してやればやっぱり (2) 式の形に帰着できる．

最後に 3 について．例えば，ある i については，$\sum_j A_{ij} x_j \geq b_i$ という不等式制約であったとする．この制約式は補助変数 s_i を導入すると以下のように書き換えることができる．

$$\sum_j A_{ij} x_j - s_i = b_i, s_i \geq 0 \ . \tag{3}$$

よって，不等式制約も (2) 式の形にまとめて書くことができる．なお，この等式が成り立つと思うと，元の不等式制約を見てやればわかるように，s_i がその不等式に関する x の「余裕度」を表している．

以上より，(2) 式は LP の一般的な形であり，これを解くことができる simplex 法は LP の一般的な解法の一つと言える．

1.3 基底解, 実行可能基底解

(今回は出てくる行列がとりあえず full rank だと思って話をします... そうでない場合はまたいづれ...)

この subsection でも，これまで通り，変数の数を n，制約式の数を m とする．勿論，これまで述べたように $n > m$ とする．

(2) の制約式 $Ax = b$ について考える．$n > m$ であるから，解くことはできない．が，n 個の変数のうち $n - m$ 個を選び，その変数を 0 としてやれば $Ax = b$ を満たすような解を得ることができる．このような解を基底解と呼ぶ．ただし，基底解は $Ax = b$ しか見ていないので，$x \geq 0$ を満たすかど

うかは不明である．基底解のうちで $x \geq 0$ も満たすようなものを実行可能基底解と呼ぶ.

以上が基底解と実行可能基底解の言葉での定義となってしまうが, simplex 法の説明にも使うので，数式でも説明しておく．

まず，変数の index 集合 $\{1, \cdots, n\}$ を m 個と $n-m$ 個の集合に分割する．前者を B_{index}, 後者を N_{index} と呼ぶことにする[3]．すると，

$$b_i = \sum_j A_{ij} x_j = \sum_{j \in B_{index}} A_{ij} x_j + \sum_{j \in N_{index}} A_{ij} x_j \tag{4}$$

と書くことができる．この右辺を以下のように書くことにする．

$$b = B x_B + N x_N \ . \tag{5}$$

これは, A の列を適当に並び替えた上で $A = [B|N] (B \in \mathbb{R}^{m \times m}, N \in \mathbb{R}^{m \times n-m})$ と分割し，さらに x を適当に並び替えて $x = (x_B, x_N)^T (x_B \in \mathbb{R}^m, x_N \in \mathbb{R}^{n-m})$ と分割して

$$b = Ax = \left[\begin{array}{c|c} B & N \end{array} \right] \left[\begin{array}{c} x_B \\ x_N \end{array} \right] = B x_B + N x_N \tag{6}$$

と考えれば同じものであることが確認できる．

このような分割 (6) を考えれば，基底解は $x_N = 0$ で特徴付けられるので，基底解は

$$x_B = B^{-1} b, x_N = 0 \tag{7}$$

と書くことができる．さらに，$x_B \geq 0$ つまり $B^{-1} b \geq 0$ が成立する場合（あるいは成立するような分割の場合）にそれを実行可能基底解と呼ぶ.

1.3.1 基底解，実行可能基底解の例

ここまで，LP の一般的な形と基底解，実行可能基底解と言葉ばかり並べてきたので，一つ例を紹介する．

以下のような制約を考えてみよう．

$$\begin{aligned} 3x_1 + 2x_2 &\leq 12 \\ x_1 + 2x_2 &\leq 8 \\ x_i &\geq 0 \ . \end{aligned} \tag{8}$$

[3]Basic と Non-Basic の略．simplex 法を見ると，個人的には Non-Basic の方が Basic な感じがするが...

こいつをイコール制約に直すと

$$\begin{aligned} 3x_1 + 2x_2 + s_1 &= 12 \\ x_1 + 2x_2 + s_2 &= 8 \\ x_i, s_i &\geq 0 \end{aligned} \tag{9}$$

となる．これを行列表記に直すと

$$\begin{bmatrix} 3 & 2 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} 12 \\ 8 \end{bmatrix} \tag{10}$$

となるので，

$$A = \begin{bmatrix} 3 & 2 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 12 \\ 8 \end{bmatrix} \tag{11}$$

であることがわかる．

この系は変数の数 $n = 4$ であり，制約式の数 $m = 2$ であるので，基底解を作るための 0 に選べる変数の数は $n - m = 2$ 個だから基底解は ${}_4C_2 = 6$ 通りだけある．

1. $N_{index} = \{x_1, x_2\}, B_{index} = \{s_1, s_2\}$

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, N = \begin{bmatrix} 3 & 2 \\ 1 & 2 \end{bmatrix} \tag{12}$$

このとき，$x_1 = x_2 = 0$ であるから，$s = b$ となる．元の変数の空間 (x_1, x_2) で見れば原点である．図からも s が「余裕度」であることがわかると思う．

2. $N_{index} = \{x_1, s_1\}, B_{index} = \{x_2, s_2\}$

$$B = \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix}, N = \begin{bmatrix} 2 & 0 \\ 2 & 1 \end{bmatrix} \tag{13}$$

このとき，基底解は $(x_1, s_1) = x_B = B^{-1}b = (8, -12)^T$ となっている．勿論 $x_2 = s_2 = 0$ である．これを元の (x_1, x_2) 空間で考えてみる．まず $x_2 = 0$ である．さらに，$s_2 = 0$ なので (8) の二つ目の不等式は等式となっている．そのため，この基底解は $x_2 = 0$ と $x_1 + 2x_2 = 8$ との交点に対応している．(以降，同じような話の場合は答えだけ書く．)

3. $N_{index} = \{x_1, s_2\}, B_{index} = \{x_2, s_1\}$

$$B = \begin{bmatrix} 3 & 0 \\ 1 & 1 \end{bmatrix}, N = \begin{bmatrix} 2 & 1 \\ 2 & 0 \end{bmatrix} \tag{14}$$

このとき，基底解は $(x_1, s_2) = x_B = B^{-1}b = (4, 4)^T$ となっている．勿論 $x_2 = s_1 = 0$ である．

4. $N_{index} = \{x_2, s_1\}, B_{index} = \{x_1, s_2\}$

$$B = \begin{bmatrix} 2 & 1 \\ 2 & 0 \end{bmatrix}, N = \begin{bmatrix} 3 & 0 \\ 1 & 1 \end{bmatrix} \tag{15}$$

このとき，基底解は $(x_2, s_1) = x_B = B^{-1}b = (4, 4)^T$ となっている．勿論 $x_1 = s_2 = 0$ である．

5. $N_{index} = \{x_2, s_2\}, B_{index} = \{x_1, s_1\}$

$$B = \begin{bmatrix} 2 & 0 \\ 2 & 1 \end{bmatrix}, N = \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} \tag{16}$$

このとき，基底解は $(x_2, s_2) = x_B = B^{-1}b = (6, -4)^T$ となっている．勿論 $x_1 = s_1 = 0$ である．

6. $N_{index} = \{s_1, s_2\}, B_{index} = \{x_1, x_2\}$

$$B = \begin{bmatrix} 3 & 2 \\ 1 & 2 \end{bmatrix}, N = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{17}$$

このとき，基底解は $(x_1, x_2) = x_B = B^{-1}b = (2, 3)^T$ となっている．勿論 $s_1 = s_2 = 0$ である．これは (x_1, x_2) 平面上で (8) の二つの不等式が共に等号が成り立つ線の交点に対応しているが，それは $s_1 = s_2 = 0$ と符号している．

この例からもわかるように，実行可能基底解は，元の空間で見ると，不等式制約が定める凸集合の端点に相当している．

2 simplex 法

準備が整ったので，LP の一般的な解法である simplex 法について述べる．ここで見るように，simplex 法は「実行可能基底解を input として，最適な実行可能基底解を output する」algorithm である．これを聞くと「input となる実行可能規定解はどう得るんだ？」と思う人も多いかと思うが，それについての答えは次の二段階 simplex の section で述べる．(この状況ではちょっと tortological に聞こえると思うが，input の実行可能基底解を作るのにも simplex を使うため．)

2.1 simplex 法の気分

今回考えている問題は線形な問題である．ということは，必ず「領域の端」で最適解を取るはずである．つまり，前 section の言葉を使えば「実行可能基底解のいずれかが，最適解を与える」となる．

さらに，良いことに，今回の objective は線形なのである．つまり，local minimum が存在しない．そのため,「今見ている実行可能基底解の『隣』の実行可能基底解で，objective が下がるものを探し続ければ最適解に辿りつく」という戦略が思いつく．そして，実は simplex 法が行っていることはほぼこれである．以下では，simplex 法の一般的な algorithm を述べた上で，この気分が正しいことを例 (といっても添付の jupyter notebook だが) で確認する．

2.2 simplex 法

2.2.1 最適性の条件, pricing rule

ある実行可能基底解 $x_B = B^{-1}b \geq 0, x_N = 0$ が与えられたとする．このときに，この解の近くで，制約を満たすように x_N を non-zero にしていくことを考える．そうした場合に，やっぱり $x_N = 0$ が objective を最小に与える，つまり，与えられた実行可能基底解が最適である条件を考える．

一般的な線形計画問題は

$$\begin{aligned} \min_x \quad & c^T x \\ s.t. \quad & Ax = b,\ x \geq 0, x \in \mathbb{R}^n \end{aligned} \tag{18}$$

こうであるが，与えられた実行可能基底解を基礎として，制約式を分解すると

$$Bx_B + Nx_N = b \Leftrightarrow x_B = B^{-1}(b - Nx_N) \tag{19}$$

となるので，これを元の問題に代入すると,objective が

$$c^T x = c_B^T x_B + c_N^T x_N = c_B^T B^{-1}(b - Nx_N) + c_N^T x_N \tag{20}$$

ということに注意すると

$$\begin{aligned} \min_{x_N} \quad & c_B^T B^{-1} b + (c_N - N^T B^{-1T} c_B)^T x_N \\ s.t. \quad & B^{-1}(b - Nx_N) \geq 0, x_N \geq 0 \end{aligned} \tag{21}$$

という x_N だけの問題に落とすことができる．ちなみに，objective の第一項は $x_N = 0$ の場合，つまり与えられた実行可能基底解の objective の値に他ならない．

今後使うので，vector

$$\rho = c_N - N^T B^{-1T} c_B \tag{22}$$

を定義しておく．勿論 (21) の第二項である．ここで，特に $x_N \geq 0$ と この ρ に注目してみる．もし $\rho > 0$ だとしよう．すると，(21) は明らかに $x_N = 0$ のときに最小値を取ることがわかる．つまり，この場合に，与えられた実行可能基底解が最適で，最適値 $c_B^T B^{-1} b$ を取る．

それに対して $\rho_j < 0 (j \in N_{index})$ となるような j があった場合はどうかというと，そのような j に対して，x_j を 0 から大きくすることで objective を下げることができる．つまり，このようなケースでは，与えられた実行可能基底解が最適ではなく，もっと objective を下げることができる．

2.2.2 pricing, ratio test

前 subsubsection で $\rho_j < 0 (j \in N_{index})$ となるような j があった場合は，例えばそのような j から一つ選び，x_j を大きくすると objective を下げることができることを確認した．以下ではそのような変数を一つ選んだ場合にどうなるかをみる．なお，このような変数の選択を pricing とか pricing rule とか呼ばれている[4]．

このように選択された j を $k \in N_{index}$ と書くことにしよう．x_N として，k にしか成分がないような状況を考える．$x_k = \xi$ と書けば， (21) は

$$\begin{aligned} \min_\xi \quad & c_B^T B^{-1} b + \rho_k \xi \\ s.t. \quad & \sum_\ell (B^{-1})_{j\ell} b_\ell \geq \sum_\ell (B^{-1})_{j\ell} A_{\ell k} \xi \ (j \in B_{index}), \\ & \xi \geq 0 \end{aligned} \tag{23}$$

と書き換えることができる．勿論，これは x_k しか動かしていないローカルな最適化問題で，もとの最適化問題とは異なる．

このローカルな最適化問題を考えるが，実はこれは簡単に解くことができる．objetive を見れば ξ は大きければ大きいほど良いが，制約式を見ると頭打ちに合っていることがわかる．具体的には

$$\bar{b}_j = \sum_\ell (B^{-1})_{j\ell} b_\ell > 0, \ y_j = \sum_\ell (B^{-1})_{j\ell} A_{\ell k} \tag{24}$$

と置いた場合に以下の θ までは増やすことができる[5]．

$$j_{min} = \arg\min(\bar{b}_j / y_j | y_j > 0), \ \theta = \bar{b}_{j_{min}} / y_{j_{min}} \tag{25}$$

勿論 $\bar{b} > 0$ は実行可能基底解を持ってきていることによる．よって，ローカルな最適化問題は $\xi = \theta$ のときに最適解を取る．

[4]pricing rule には，一応色々な方法が考案されている．

[5]なお，もし全て $y_i < 0$ ならば，このローカルな問題，ひいては元々の問題の答えは非有界である．

さて，与えられた実行可能基底解の周りについて考えていたが，この実行可能基底解で x_N としてこの k の成分が θ になったときどうなるか，というと $x_B^{mod} = B^{-1}(b - Nx_N)$ に代入して計算すると，

$$x_B^{mod}{}_j = \bar{b}_j - y_j\theta \tag{26}$$

となるが, θ の選び方から, $x_B^{mod}{}_j > 0 (j \neq j_{min})$ かつ $x_B^{mod}{}_{j_{min}} = 0$ である．勿論 $x_k = \theta \neq 0$ である．よって, ローカルな最適化問題の最適解は, 元の実行可能基底解の B_{index} と N_{index} を利用して書けば, $B_{index}^{new} = (B_{index} \backslash \{j_{min}\}) \cup \{k\}$, $N_{index}^{new} = (N_{index} \backslash \{k\}) \cup \{j_{min}\}$ で書かれる実行可能基底解である．勿論この新しい実行可能基底解の objective は元の実行可能規定解の objective より小さな値を取っている．さらに，新しいものも実行可能基底解である．よって同じ操作，つまりローカルな問題 (23) を定義して解くこと，を繰り返すことでどんどん objective を下げていくことができる．そして，LP なので，このように下げていっても local minimum にはまることはないので，これで最適解に辿りつくことができるというわけである．

これが simplex 法の根本的な考え方であるが，念のため一つだけ指摘をしておくと，実行可能基底解が与えられ pricing で k を選んだのちのローカルな最適化問題 (23) であるが，この問題自体は (24) を計算して (25) を確認するだけの簡単な問題である．特に (25) から ratio test と呼ばれてる作業だが，ほぼ簡単な代数操作をするだけである．よって simplex 法は突き詰めれば，「pricing と ratio test を繰り返す」だけの algorithm とも言える．

2.2.3 algorithm

以上まとめると simplex 法の algorithm は以下の通りである．

2.3 例

以下の問題を考える．

これを解いていく過程が添付の jupyter notebook となっている．これを見ていただければ，

3 二段階 simplex 法

以上に見たように simplex 法は「実行可能基底解を input として，最適実行可能基底解を output する」algorithm であった．ここで勿論気になるのは「input である実行可能規定解をどう作るのか」である．実は，この input も simplex 法で作ることができる．

Require: 実行可能基底解
Ensure: 最適な実行可能基底解
現在の実行可能規定解 ← inpout 実行可能基底解
loop
現在の実行可能基底解と (22) に基づいて ρ を計算する.
if $\rho > 0$ **then**
return 現在の実行可能基底解解
end if
– pricing –
なんらかのルールで pricing を行って k を選ぶ.
– ratio test –
(24) を計算
(25) によって $B_{index} \to N_{index}$ となる変数 j_{min} を選択
– 実行可能基底解の更新 –
k, j_{min} を基に現在の実行可能基底解を更新
end loop

まずは，毎度お馴染み LP の一般的な問題からスタートする.

$$\begin{aligned} \min_x \quad & c^T x \\ s.t. \quad & \sum_j A_{ij} x_j = b_i \ (i \in \{1, \cdots, m\}), \\ & x \geq 0, x \in \mathbb{R}^n \end{aligned} \tag{27}$$

この問題に対して以下のような問題を考えてみよう.

$$\begin{aligned} \min_{x,s} \quad & \sum_{i=1}^{m} s_i \\ s.t. \quad & \sum_j A_{ij} x_j + sign(b_i) s_i = b_i \ (i \in \{1, \cdots, m\}), \\ & x \geq 0, s \geq 0, x \in \mathbb{R}^n, s \in \mathbb{R}^m \ , \end{aligned} \tag{28}$$

ここに $sign(a) = a/|a|$ である．この問題の意味は次の通りである.

(28) の気分

x が何か与えられたときに，各制約について s はその破れ具合を表している．今回の objective は破れ具合の和 $\sum_i s_i$ であるから，この問題を解き，その結果が objective $= 0$ な解だった場合には，元問題の実行可能解が得られる.

(28) 式 を simplex 法で解いてみよう．simplex 法は「実行可能基底解を input として，最適実行可能基底解を output する」algorithm であった．(28) 式はありがたいことに，以下の自明な実行可能基底解が存在する．

$$x_j = 0, s_i = |b_i| \ . \tag{29}$$

つまり，(28) 式の simplex 法の input は問題なく用意できる．なので，こいつを input にして simplex 法を回すことができる．その output は何かと言うと (28) 式の最適な実行可能基底解である．さてその解であるが，(28) の目的関数が元問題の制約式の破れ具合の $\sum_i s_i$ であることから，もし (27) が infeasible でなかった場合は，(28) の最適解 (x, s) は $s_i = 0$ であるような実行可能基底解であるはずである．そして $s_i = 0$ であることから，(28) を見ればわかるように，そのような実行可能基底解は，元問題 (27) の実行可能基底解となっている．よって (28) を simplex 法で解くことによって (27) の input を作成することができる．

以上のように，input となる実行可能基底解 も simplex 法で作ることができるので，結果的に二回 simplex 法を解くことで一般的な LP を解くことが可能である．そのためこのような解き方を 二段階 simplex 法と呼ばれている．まとめれば，二段階 simplex 法を利用することで一般的な LP を input として，最適解を output できる．

4 まとめ

この資料では，一般的な LP や LP にまつわる用語から始め，simplex 法や二段階 simplex 法について解説をした．しかし，

- 実は simplex はちょっと遅い (内点法の法が一般的には速い)．
- また，最適解を切り落とすような制約式の追加に弱い (dual simplex であれば問題ない．この性質があるために MILP の一般的解法である branch and bound では dual simplex が使われている．)．

といった問題がある．これらの解決は今後このゼミでなされ続けるはずである．