Pesquisa Operacional

Igor M. Coelho

6 de Julho de 2020

Section 1

Método Simplex

Sobre esse material

Esses slides foram possíveis devido a contribuições de diversas pessoas/materiais, em especial:

- Notas do prof. Marcone Jamilson Freitas Souza
- Livro Nelson Maculan e Marcia Fampa
- Livro-texto do curso
- [2] Tutorial ilectures (https://igormcoelho.github.io/ilectures-pandoc/)
- Minha esposa Cristiane Tavares pelas valiosas dicas na elaboração desse material

Fundamentos Necessários

Caso não se sintam confiantes nos tópicos abaixo, façam uma revisão antes de aprofundar neste material:

- Álgebra Linear e Geometria Analítica
  - Equação da Reta
  - Operações Vetoriais
  - Operações com Matrizes
- Métodos Numéricos
  - Método da Eliminação de Gauss
- Cálculo
  - Gradientes e Otimização

Section 2

História do Simplex

Breve história

De acordo com Maculan&Fampa (2006)[1], as primeiras ideias de como otimizar um sistema de desigualdades lineares foi explorado por Fourier[2] em 1880, porém somente George Dantzig[3] em 1947 que de fato propôs o método de resolução *simplex*.

O simplex é um algoritmo reconhecidamente bem-sucedido, tendo sido implementado em diversos *solvers* de computador altamente eficientes, como CPLEX, Gurobi, CBC (*open-source*), etc.

---

[1] N. Maculan e M. Fampa. *Otimização Linear*. Editora UnB, 2006.

[2] Fourier, J.B.J. *Oeuvres*. "Second Extrait", G. Darboux, Gauthiers-Villars, p. 325-328, 1880.

[3] Dantzig, George B. Maximization of a linear function of variables subject to linear inequalities. *Activity Analysis of Production and Allocation*. In: KOOPMANS, C. (Ed.). New York: Wiley, p. 359-373, 1951.

Aplicações no planejamento da produção e outros métodos

Em 1939, L. Kantorovich[4] modelou e resolveu matematicamente problemas de planejamento da produção na União Soviética, ganhando o prêmio Nobel de Economia em 1975.

Outros métodos para resolução: Métodos Elipsoidais de L. Khachian[5] em 1978; Métodos de Pontos Interiores de N. Karmarkar[6] em 1984; embora elegantes (com garantia de tempo polinomial), <u>são tipicamente menos eficientes</u> *na prática* que o simplex.

[4] Kantorovich, L. *Métodos Matemáticos na Organização e no Planejamento da Produção* (em russo). Leningrado: Editora da Univ. Estatal de Leningrado, 1939 (tradução inglesa: *Management Science*, v.6, p. 366-422, 1960).

[5] Khachian, L. A polynomial algorithm for linear programming. Doklady Academiia Nauk SSSR, v.244, p.191-194 (em russo. Tradução em inglês: *Soviet Mathematics Doklady*, v.20, p.191-194, 1979.).

[6] Karmarkar, N. A new polynomial algorithm for linear programming. *Combinatorica*, v.4, p.373-395, 1984.

Section 3

Fundamentos do Simplex

Um Problema de Programação Linear

Considere o seguinte problema de programação linear:

$$maximizar\ f(x) = \sum_{j=1}^{p} c_j x_j$$

Sujeito a:

$$\sum_{j=1}^{p} a_{ij} x_j \leq b_i,\ \ i = 1, 2, ..., q$$

$$x_j \geq 0,\ \ j = 1, 2, ..., p$$

onde: c_j, a_{ij} e b_i são dados (números reais) e x_j representa as variáveis de decisão (não-negativas). Consideramos, neste caso, uma função objetivo $f(x)$ de maximização, e restrições do tipo $\leq$.

Variáveis de Folga

Restrições do tipo $\leq$ (ou $\geq$) podem ser facilmente transformadas em igualdades, com a introdução de novas variáveis (não-negativas) de folga/falta (do inglês, *slack*/*surplus*):

$$\sum_{j=1}^{p} a_{ij}x_j \leq b_i \iff \begin{cases} \sum_{j=1}^{p} a_{ij}x_j + x_{p+1} = b_i, \\ x_{p+i} \geq 0 \end{cases}$$

$$\sum_{j=1}^{p} a_{ij}x_j \geq b_i \iff \begin{cases} \sum_{j=1}^{p} a_{ij}x_j - x_{p+1} = b_i, \\ x_{p+i} \geq 0 \end{cases}$$

Variáveis de Folga (exemplo)

Um exemplo de transformação de $\leq$ em igualdade (introduzindo variável de folga x_3):

$$2x_1 + 3x_2 \leq 5 \Rightarrow 2x_1 + 3x_2 + \underbrace{x_3}_{x_3 \geq 0} = 5$$

O mesmo para restrições $\geq$ (introduzindo variável x_4):

$$x_1 + 6x_2 \geq 7 \Rightarrow x_1 + 6x_2 - \underbrace{x_4}_{x_4 \geq 0} = 7$$

Outras conversões à forma padrão

Demais técnicas de conversão de variáveis/restrições:

- Existe $b_i < 0$:
  - **Solução:** multiplique a restrição i por -1
- Existem variáveis não positivas (seja $x_k \leq 0$):
  - **Solução:** Substituir por variável $x'_k \geq 0$ tal que $x'_k = -x_k$
- Existem variáveis livres $x_k \gtrless 0$ (seja $x_k \in \mathbb{R}$):
  - **Solução:** substituir x_k por $x'_k - x''_k$, tal que $x'_k \geq 0$ e $x''_k \geq 0$
- Um problema de minimização pode ser convertido em maximização (vice-versa):

$$maximizar\ f(x) = -minimizar\ \{-f(x)\}$$

Tipos de PPL

Listamos três tipos fundamentais de PPL[7]:

- padrão (*standard*): $Ax = b$ e $x \geq 0$
- canônico (*canonical*): $Ax \geq b$ e $x \geq 0$
- geral (*general*): $a_i x = b_i$ ($\forall\ i \in M$), $a_i x \geq b_i$ ($\forall\ i \in \bar{M}$), $x_j \geq 0$ ($\forall\ j \in N$), $x_j \gtrless 0$ ($\forall\ j \in \bar{N}$)

[7] Christos Papadimitriou & Kenneth Steiglitz. *Combinatorial Optimization*, 1982 (1998).

Problema de Programação Linear Padrão

Sempre poderemos escrever um **problema de programação linear na forma padrão** (PPL):

$$(PPL): \ maximizar \ f(x) = \sum_{j=1}^{n} c_j x_j$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \ i = 1, 2, ..., m$$

$$x_j \geq 0, \ j = 1, 2, ..., n$$

tendo assim, n variáveis e m restrições.

Problema de Programação Linear Padrão (vetores)

De forma equivalente, podemos representar o PPL na forma vetorial:

$$(PPL): \ maximizar \ z = cx$$
$$Ax = b$$
$$x \geq 0$$

onde $c = (c_1 \ c_2 \ ... \ c_{\mathcal{N}})$, $x^\intercal = (x_1 \ x_2 \ ... \ x_{\mathcal{N}})$, $b^\intercal = (b_1 \ b_2 \ ... \ b_m)$, $A = (a_1 \ a_2 \ ... \ a_n)$ e $a_j^\intercal = (a_{1j} \ a_{2j} \ ... a_{mj})$, isto é, $c^\intercal \in \mathbb{R}^n$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ e $a_j \in \mathbb{R}^m$.

Definição 2.1 (Maculan&Fampa) Seja $X = \{x \in \mathbb{R}^n | Ax = b, x \geq 0\}$ a **região viável** do PPL, e $x \in X$ uma **solução viável** do PPL. Se $x^* \in X$ tal que $cx^* \geq cx, \forall x \in X$, x^* é uma **solução ótima**.

Exemplo de PPL

$$\begin{array}{lccc} max & x_1 & +2x_2 & \\ & x_1 & & \leq 2 \\ & & x_2 & \leq 2 \\ & x_1 & +x_2 & \leq 3 \\ & x_1, & x_2 & \geq 0 \end{array} \Rightarrow \begin{array}{lcccccc} max & x_1 & +2x_2 & +0x_3 & +0x_4 & +0x_5 & \\ & x_1 & & +x_3 & & & = 2 \\ & & x_2 & & +x_4 & & = 2 \\ & x_1 & +x_2 & & & +x_5 & = 3 \\ & x_1, & x_2, & x_3, & x_4, & x_5 & \geq 0 \end{array}$$

$$\underbrace{\begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}}_{b}$$

Matriz básica e não-básica

A matriz $A_{m \times n}$ pode ser particionada da seguinte maneira (supondo $posto(A) = m$, com m colunas independentes):

$$A = (B \; N)$$

onde $B_{m \times m}$, chamada de matriz básica, é inversível; e $N_{m \times (n-m)}$ é chamada de não-básica. Analogamente, particionamos x e c, tal que: $x^\intercal = (x_\mathcal{B}^\intercal \; x_\mathcal{N}^\intercal)$, $c = (c_\mathcal{B} \; c_\mathcal{N})$. Vetores $x_\mathcal{B}$ e $c_\mathcal{B}$ possuem m componentes associadas à matriz B. Reescrevemos o PPL:

$$(PPL): \; maximizar \; z = c_\mathcal{B} x_\mathcal{B} + c_\mathcal{N} x_\mathcal{N}$$

$$B x_\mathcal{B} + N x_\mathcal{N}$$

$$x_\mathcal{B} \geq 0, x_\mathcal{N} \geq 0$$

Solução básica e não-básica

Explicitamos $x_{\mathcal{B}}$ em função de $x_{\mathcal{N}}$ (Eq. 2.10 em Maculan&Fampa):

$$x_{\mathcal{B}} = B^{-1}b - B^{-1}Nx_{\mathcal{N}}$$

Faremos $x_{\mathcal{N}} = 0$ e $\bar{x}_{\mathcal{B}} = B^{-1}b$.

Definição 2.2 (Maculan&Fampa) $\bar{x}$ é uma **solução básica**, se $\bar{x}^\intercal = (\bar{x}_{\mathcal{B}}^\intercal \; 0)$.

Quando $\bar{x}_{\mathcal{B}} \geq 0$, será uma **solução básica viável**.

Sejam $I_{\mathcal{B}}$ o conjunto dos índices das colunas de A pertencendo à matriz B, e $I_{\mathcal{N}}$ os demais índices de A, tal que: $I_{\mathcal{B}} \cap I_{\mathcal{N}} = \varnothing$ e $I_{\mathcal{B}} \cup I_{\mathcal{N}} = \{1, 2, ..., n\}$.

PPL com matriz básica

$$(PPL): \ \ maximizar \ z = c_{\mathcal{B}}B^{-1}b - (c_{\mathcal{B}}B^{-1}N - c_{\mathcal{N}})x_{\mathcal{N}}$$

Sujeito a:

$$x_{\mathcal{B}} = B^{-1}b - B^{-1}Nx_{\mathcal{N}}$$

$$x_{\mathcal{B}} \geq 0, x_{\mathcal{N}} \geq 0$$

PPL com notação aprimorada

De acordo com Maculan&Fampa, definiremos:

- $\lambda = c_{\mathcal{B}} B^{-1}$, $\lambda^{\intercal} \in \mathbb{R}^m$
- $\bar{x}_{\mathcal{B}} = B^{-1} b$, $\bar{x}_{\mathcal{B}} \in \mathbb{R}^m$
- $z_j = \lambda a_j$, $(j \in I_{\mathcal{B}} \cup I_{\mathcal{N}})$, $z_j \in \mathbb{R}$
- $y_j = B^{-1} a_j$, $(j \in I_{\mathcal{B}} \cup I_{\mathcal{N}})$, $y_j \in \mathbb{R}^m$
- $\bar{z} = c_{\mathcal{B}} B^{-1} b = \lambda b = c_{\mathcal{B}} \bar{x}_{\mathcal{B}}$.

Então teremos um novo PPL aprimorado:

$$(PPL): \ maximizar \ z = \bar{z} - \sum_{j \in I_{\mathcal{N}}} (z_j - c_j) x_j$$

sujeito a:

$$x_{\mathcal{B}} = \bar{x}_{\mathcal{B}} - \sum_{j \in I_{\mathcal{N}}} y_j x_j$$

Otimalidade no PPL

Proposição 2.1 (de Maculan&Fampa) Se $\bar{x}_{\mathcal{B}} \geq 0$ e $z_j - c_j \geq 0$, $\forall j \in I_{\mathcal{N}}$, então o vetor $x^* \in \mathbb{R}^n$, onde $x^*_{\mathcal{B}(i)} = \bar{x}_{\mathcal{B}(i)}$, $i = 1, 2, ..., m$ e $x^*_j = 0$, $j \in I_{\mathcal{N}}$, será uma solução ótima do `(PPL)`.

Focaremos agora na versão do Simplex por tabelas, após apresentar um pseudo-código do algoritmo (com base no livro-texto de Arenales).

Section 4

Detalhamento do Simplex

Simplex para problemas de $\leq$

O Simplex consiste de duas fases, onde a primeira consiste em encontrar uma base B.

Para problemas com restrições $\leq$, as variáveis de folga introduzidas no modelo irão naturalmente formar uma matriz identidade $\mathcal{I}_m$.

Assim, escolheremos essas variáveis de folga como *variáveis básicas*, atribuindo valor zero a todas as demais *variáveis não-básicas* (originais do modelo). Teremos assim uma base inversível $B = \mathcal{I}_m$. Neste caso, a primeira fase do Simplex já é naturalmente efetuada.

Pseudo-código do Simplex (Fase II) livro-texto

1. Passo 1: cálculo da solução básica

$$\begin{cases} \bar{x}_{\mathcal{B}} = B^{-1}b \\ \bar{x}_N = 0 \end{cases}$$

2. Passo 2: cálculo dos custos relativos
   1. Vetor multiplicador simplex
      - $\lambda^\intercal = c_{\mathcal{B}}^\intercal B^{-1}$
   2. Custos relativos
      - $\hat{c}_{N(j)} = c_{N(j)} - \lambda^\intercal a_{N(j)},\ j = 1, 2, ..., n - m$
   3. Determinação de variável a entrar na base
      - $\hat{c}_{N(k)} = min\{\hat{c}_{N(j)}, j = 1, ..., n - m\}$ (a variável $x_{N(k)}$ entra na base)
3. Passo 3: teste de otimalidade (minimização)
   - Se $\hat{c}_{N(k)} \geq 0$, então: *pare* (solução atual é ótima!).

Pseudo-código do Simplex (Fase II) livro-texto

4. Passo 4: Cálculo da direção Simplex
   - $y = B^{-1} a_{\mathcal{N}(k)}$
5. Passo 5: Determinação do passo e variável a sair da base
   - Se $y \leq 0$, então: *pare* (não existe solução ótima finita: $f(x) \rightarrow -\infty$)
   - Caso contrário, determine a variável a sair da base pela razão mínima:

$$\hat{\varepsilon} = \frac{\hat{x}_{\mathcal{B}(\ell)}}{y_\ell} = min \left\{ \frac{\hat{x}_{\mathcal{B}(i)}}{y_i} : y_i > 0, i = 1, ..., m \right\}$$

     (variável $x_{\mathcal{B}(\ell)}$ sai da base)
6. Passo 6: atualização
   - matriz básica: $\mathcal{B} = (a_{\mathcal{B}(1)} \ldots a_{\mathcal{B}(\ell-1)} \; a_{\mathcal{N}(k)} \; a_{\mathcal{B}(\ell+1)} \ldots a_{\mathcal{B}(m)})$
   - não-básica: $\mathcal{N} = (a_{\mathcal{N}(1)} \ldots a_{\mathcal{N}(k-1)} \; a_{\mathcal{B}(\ell)} \; a_{\mathcal{N}(k+1)} \ldots a_{\mathcal{N}(n-m)})$
   - incrementa iteração e volte ao Passo 1

Section 5

Exemplo com Python

Exemplo do Simplex

Vide "Exemplo 2.26" do livro-texto de Arenales (página 85).

$$
\begin{array}{lccl}
\textit{minimizar } f(x_0, x_1) = & -x_0 & -2x_1 & \\
& x_0 & +x_1 & \leq 6 \\
& x_0 & -x_1 & \leq 4 \\
& -x_0 & +x_1 & \leq 4 \\
& x_0, & x_1 & \geq 0
\end{array}
$$

Solução Básica Ótima: $x_{\mathcal{B}} = (x_0, x_3, x_1)$, tal que $f(x_{\mathcal{B}}) = -11$

Exemplo com Python (dados do problema)

Primeiramente, adicionamos restrições de folga $\leq$ (novas variáveis x_2, x_3 e x_4), e obtemos uma matriz identidade $\mathcal{I}_3$ como base B para o passo 1 do Simplex: $\mathcal{B} = (2, 3, 4)$.

Dados do problema:

	x_0	x_1	x_2	x_3	x_4	b
A	1	1	1	0	0	6
ˆ	1	-1	0	1	0	4
ˆ	-1	1	0	0	1	4
c	-1	-2	0	0	0	Min f

Exemplo com Python (construindo base)

```
import numpy as np
A=np.array([[1,1,1,0,0],[1,-1,0,1,0],[-1,1,0,0,1]])
b=np.array([6,4,4])
c=np.array([-1, -2, 0, 0, 0])
#
IB=[2,3,4]  # variaveis "de folga" na base
IN=[0,1]    # variaveis "originais" não-básicas
# Construindo a Base a partir das variáveis dadas
Base=np.transpose(np.asarray([A[:,IB[0]], A[:,IB[1]],
                                          A[:,IB[2]]]))
#>>> Base
#array([[1, 0, 0],
#        [0, 1, 0],
#        [0, 0, 1]])
```

Exemplo com Python (primeira iteração - passo 1)

Passo 1 - Cálculo da solução básica (resolva $B \cdot x_{\mathcal{B}} = b$) e obtenha:

$$\hat{x}_{\mathcal{B}} = \begin{pmatrix} 6 \\ 4 \\ 4 \end{pmatrix}$$

```
x = np.linalg.inv(Base).dot(b)
#>>> x
#array([6., 4., 4.])
```

Exemplo com Python (primeira iteração - passo 2)

Passo 2 - Calcule custos relativos (para N_0 e N_1):
$c_\mathcal{B} = (c_{\mathcal{B}(0)}, c_{\mathcal{B}(1)}, c_{\mathcal{B}(2)})$, $B^\intercal \lambda = c_\mathcal{B}$, onde $\lambda^\intercal = (0, 0, 0)$.

```
cB = [ c[IB[0]], c[IB[1]], c[IB[2]] ]
# calcula "lambda" (chamado 'u' aqui)
u = np.linalg.inv(np.transpose(Base)).dot(cB)
#>>> u
#array([0., 0., 0.])
```

- $\hat{c}_0 = c_0 - \lambda^\intercal a_0 = -1$

```
a0 = A[:,0]
cr0 = c[0] - u.dot(a0)
#>>> cr0
#-1.0
```

- $\hat{c}_1 = c_1 - \lambda^\intercal a_1 = -2$

```
a1 = A[:,1]
cr1 = c[1] - u.dot(a1)
#>>> cr1
#-2.0
```

($x_{B(1)} = x_1$ entra na base)

Exemplo com Python (primeira iteração - passos 3-6)

Passo 3 dispensado ($\hat{c}_1 = -2 < 0$), solução não é ótima! Vamos ao passo 4 para cálculo da direção simplex: resolva $By = a_1$ e obtenha $y^\intercal = (1 \ -1 \ 1)$.

```
y = np.linalg.inv(Base).dot(a1)
#>>> y
#array([ 1., -1.,  1.])
#>>> x
#array([6., 4., 4.])
#>>> x/y
#array([ 6., -4.,  4.])
```

Escolhemos $\hat{\varepsilon} = \frac{\hat{x}_{\mathcal{B}(2)}}{y_2} = 4$, então $x_{\mathcal{B}(2)} = x_4$ sai da base: $\mathcal{B} = (2, 3, 1)$, $\mathcal{N} = (0, 4)$, $f(x) = f(\hat{x}) + \hat{c}_{\mathcal{N}(k)}\hat{\varepsilon} = 0 - 2 \times 4 = -8$.

Exemplo com Python (segunda iteração - passos 1-6)

Passo 1 - Cálculo da solução básica (resolva $B \cdot x_{\mathcal{B}} = b$) e obtenha:

$$\hat{x}_{\mathcal{B}} = \begin{pmatrix} 2 \\ 8 \\ 4 \end{pmatrix}$$

```
IB=[2,3,1]  # variaveis na base
IN=[0,4]    # variaveis fora da base
# Construindo a Base a partir das variáveis dadas
Base=np.transpose(np.asarray([A[:,IB[0]], A[:,IB[1]],
                                              A[:,IB[2]]]))
x = np.linalg.inv(Base).dot(b)
#>>> x
#array([2., 8., 4.])
```

Avance nos passos 2-6 e obtenha: $\mathcal{B} = (0, 3, 2)$, $\mathcal{N} = (2, 4)$.

Exemplo com Python (terceira iteração)

Passo 1 - Cálculo da solução básica (resolva $B \cdot x_{\mathcal{B}} = b$) e obtenha:

$$\hat{x}_{\mathcal{B}} = \begin{pmatrix} 1 \\ 8 \\ 5 \end{pmatrix}$$

```
IB=[0,3,1]   # variaveis na base
IN=[2,4]     # variaveis fora da base
# Construindo a Base a partir das variáveis dadas
Base=np.transpose(np.asarray([A[:,IB[0]], A[:,IB[1]],
                                          A[:,IB[2]]]))
x = np.linalg.inv(Base).dot(b)
#>>> x
#array([1., 8., 5.])
```

Avance ao passo 2 e descubra que solução é ótima!

Exemplo com Python (solução ótima)

Obtenha valor $f(x) = -11$ na solução ótima $\hat{x}^T = (1, 5, 0, 8, 0)$:

```
IB=[0,3,1]   # variaveis na base
IN=[2,4]     # variaveis fora da base
# Construindo a Base a partir das variáveis dadas
Base=np.transpose(np.asarray([A[:,IB[0]], A[:,IB[1]],
                                                 A[:,IB[2]]]))
x = np.linalg.inv(Base).dot(b)
#>>> x
#array([1., 8., 5.])
cB = [ c[IB[0]], c[IB[1]], c[IB[2]] ]
#>>> sum(cB*x)
#-11.0
```

Section 6

Tableau Simplex

Simplex por Tabelas

Uma versão prática do Simplex pode ser feita com tabelas (*tableau simplex*).

No caso de não haver apenas restrições $\leq$, é necessário criar *variáveis artificiais*, bem como um novo problema de otimização que busca *minimizar* o valor delas (a zero!). Nesse PPL estendido, o peso inicial é 0 para as variáveis do PPL original, e 1 para as artificiais. Quando a otimalidade é atingida nesse modelo (e as variáveis artificiais saem da base), podemos cortar as variáveis artificiais, e retornar ao modelo original (fase 2).

Os slides do prof. Marcone detalham o passo-a-passo dessa abordagem: Slides SIMPLEX (pdf).

Lista de Exercícios

A lista de exercícios está disponibilizada no site.