Implementação dos algoritmos Primal Simplex e Dual Simplex

Relatório

Francilândio Lima Serafim
Vanessa Carvalho do Nascimento

UNIVERSIDADE
FEDERAL DO CEARÁ

12 de Julho de 2023

Conteúdo

1 Forma canônica

A forma canônica é uma representação específica de um problema de programação linear que é frequentemente usada para aplicar o algoritmo simplex. O algoritmo simplex é um método utilizado para resolver problemas de otimização em programação linear, encontrando a solução ótima,caso exista, para um conjunto de restrições e uma função objetivo.

Ao colocar um problema linear na forma canônica, ele é expresso em termos de restrições de igualdade (ao invés de desigualdades) e todas as variáveis são não negativas. Essa representação facilita a aplicação do algoritmo simplex, que opera iterativamente por meio de operações de pivoteamento.

O objetivo das operações de padronização é transformar o problema apresentado para o seguinte molde:

$$\begin{aligned} \min \quad & C^T x \\ \text{s.a.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Sendo,

1. C (Coeficientes da Função objetivo): C representa a função objetivo do problema linear. É uma expressão matemática que define a quantidade que se deseja maximizar ou minimizar. Em um problema de maximização, C geralmente é uma combinação linear das variáveis do problema, multiplicadas por coeficientes correspondentes. Em um problema de minimização, C é semelhante, mas a função objetivo deve ser minimizada.

2. x (Variáveis de decisão): x é um vetor (ou uma matriz) que representa as variáveis de decisão do problema linear. Essas variáveis são as incógnitas que devem ser determinadas para alcançar a solução ótima do problema. Cada componente do vetor x corresponde a uma variável específica.

3. A (Matriz de coeficientes das restrições): A é uma matriz que representa os coeficientes das restrições do problema linear. As restrições são equações ou desigualdades que limitam as soluções viáveis do problema. A matriz A é composta por elementos que multiplicam as variáveis de decisão em cada restrição. Cada linha da matriz A corresponde a uma restrição e cada coluna corresponde a uma variável de decisão.

4. b (Vetor de termos constantes das restrições): b é um vetor que representa os termos constantes das restrições do problema linear. Em um problema de igualdade, b é o lado direito da equação de restrição correspondente. Em um problema de desigualdade, b é o lado direito da desigualdade correspondente. Cada elemento do vetor b corresponde a uma restrição específica.

Os passos para transformar um problema linear em sua forma canônica são os seguintes:

1. Caso o objetivo seja maximizar:

   Nesse caso, troca-se o sinal dos coeficientes da função de custo.

2. Caso de restrição do tipo *menor ou igual*:

   Se, por exemplo, $x_j \leq b_i$, acrescenta-se uma nova variável p tal que $x_j + p = b_i$, sendo $p \geq 0$.

3. Caso de restrição do tipo *maior ou igual*:

   Se, por exemplo, $x_j \geq b_i$ Acrescenta-se uma nova variável p tal que $x_j - p = b_i$, sendo $p \geq 0$.

4. Caso em que x_j é uma variável livre

   Acrescenta-se duas novas variáveis p_1 e p_2, com p_1 e $p_2 \geq 0$, tal que $p_1 - p_2 = x_j$, e a partir disso faz-se as adaptações necessárias.

5. Caso em que $x_j \leq r$

   Se $r = 0$, troca-se o sinal de r.

   Se $r \neq 0$, cria-se uma nova restrição.

6. Caso em que $x_j \geq r$ e $r \neq 0$

   Cria-se uma nova variável.

2 Algoritmo Primal Simplex

2.1 Introdução

O método Simplex é um processo iterativo que permite melhorar a solução da função objetivo em cada etapa. O processo finaliza quando não é possível continuar melhorando este valor, ou seja, quando se obtenha a solução ótima (o maior ou menor valor possível, acordado com a função objetivo, para o qual todas as restrições sejam satisfeitas).

Um dos problemas dessa abordagem é achar uma solução básica viável (bfs), para que seja possível "caminhar"em uma direção que se aproxime cada vez mais do valor ótimo.

Para achar uma bfs inicial, o algoritmo Fase 1 é aplicado.

2.2 Fase 1

Para resolver o problema de achar uma bfs inicial do método simplex, algumas modificações são feitas na forma original do PPL.

O primeiro passo, considerando que o problema esteja na forma padrão, é acrescentar variáveis artificiais y ao problema original, após isso uma nova função objetivo é considerada.

$$\begin{aligned} & min \quad ey \\ sa \quad & Ax + Iy = b \\ & x, \quad y \geq 0 \end{aligned}$$

A bfs inicial da Fase 1 é definida pela matriz de coeficientes de $\mathbf{y}$, ou seja, $\mathbf{B} = \mathbf{I}$, sendo calculada a inversa de $\mathbf{B}$, $\mathbf{B}^{-1}$, para calcular o vetor coluna $\mathbf{xb}$ definido por:

$$xb = B^{-1} \cdot b$$

Ademais, são definidos vetores que representam os custos das variáveis que estão na base ($\mathbf{C}_B$) e não estão na base ($\mathbf{C}_N$), para que sejam usadas no teste de otimalidade juntamente com a matriz das colunas de $\mathbf{AI}$:

$$C_N - C_B.B^{-1}.N$$

De modo que se o valor da expressão anterior for um vetor maior ou igual a $\mathbf{0}$, o valor ótimo foi atingido, sendo verificados três possíveis casos:

- Nenhum y está na base: a bfs encontrada é aplicada na Fase 2 como inicial.
- Há pelo menos um y na base, mas todos com valor igual a zero: a bfs é usada como inicial no LP auxiliar.
- Há alguma variável y na base e seu valor é diferente de zero: O problema é inviável.

Caso não seja verificado que o vetor do teste é maior ou igual a $\mathbf{0}$, uma nova bfs é calculada a partir do pivoteamento entre colunas de $\mathbf{B}$ e $\mathbf{N}$.

Para a escolha da variável que entra na base, com objetivo de evitar o problema de solução degenerada é aplicada a regra de Bland, que seleciona a coluna j de $\mathbf{N}$ tal que $(C_N - C_B.B^{-1}.N)_{.j} < 0$ e j é a coluna mais a esquerda em $\mathbf{AI}$.

O valor de j é usado no cálculo da direção de movimento **db**, da seguinte forma:

$$db = -B^{-1}.N_{.j}$$

Sabendo que o algoritmo Fase 1 é limitado, não é preciso fazer testes sobre o valor de **db**, logo podendo ser obtida a variável que deve sair da base. Tal procedimento é feito de forma a encontrar

$$t = min\{\frac{-xb_k}{db_k}, db_k < 0\}$$

Onde k é o índice da variável que sai da base, ou seja, a coluna k em **B** será trocada com a coluna j em **N**, iniciando novamente o algoritmo Fase 1.

2.3 Fase 2

A Fase 2 é aplicada quando uma bfs ótima é encontrada na Fase 1 e nenhuma das variáveis artificiais y está na base, sendo essa a bfs inicial do problema original.

As etapas dessa Fase se diferem da Fase 1 em poucos aspectos, nela o objetivo é achar o valor ótimo da função objetivo caminhando na direção apropriada a partir da bfs inicial calculada, podendo-se também chegar a conclusão de que o problema é ilimitado (quando $\mathbf{db} \geq \mathbf{0}$).

2.4 Problema Auxiliar

O algoritmo LP auxiliar é aplicado quando uma bfs é encontrada na Fase 1 porém não pode ser aplicada na Fase 2, pois alguma variável artificial y está na base mas com valor nulo.

A partir da bfs fornecida, algumas modificações são feitas na forma do problema, uma dessas modificações é o acréscimo de uma variável z, ficando o problema como a seguir:

$$\begin{aligned} & & min \quad & cx \\ sa \quad & Ax + Iy + \mathbf{0}z = b \\ & & & ey + z = 0 \\ & x, \quad & y, \quad & z \geq \mathbf{0} \end{aligned}$$

Assim como a Fase 2, o LP auxiliar repete as etapas da Fase 1 com a diferença de estar buscando o valor ótimo da função objetivo a partir da bfs fornecida na Fase 1, caminhando na direção adequada para o problema e podendo chegar a conclusão de que o problema é ilimitado ($\mathbf{db} \geq \mathbf{0}$)

2.5 Dificuldades de implementação

As principais dificuldades na implementação do simplex primal dizem respeito às operações matriciais, pois apesar do uso de uma linguagem que apresenta uma biblioteca completa para manipulação de matrizes (Python - Numpy), foi preciso se habituar aos comandos, ao longo da execução dos códigos apareceram muitos problemas relativos às ordens das matrizes, problemas com o fato de as matrizes serem passadas por referência, sendo necessário usar sempre cópias para não afetar sua instância original.

As dificuldades de implementação acabaram agregando mais conhecimento sobre o poder da linguagem Python, abordagens que facilitaram os cálculos como indexação avançada foram úteis ao mesmo tempo que novas.

3 Algoritmo Dual Simplex

3.1 Introdução

O método Dual Simplex é uma variante do algoritmo Simplex que é usado para resolver problemas de programação linear na forma dual. Ele é especialmente útil quando o problema está na forma padrão e possui restrições de igualdade.

O objetivo do Dual Simplex é encontrar uma solução ótima ao otimizar a função objetivo dual. Para entender melhor o método, vamos considerar um problema de programação linear na forma padrão com a função objetivo primal $min : c^T x$ e as restrições $Ax = b$, onde c é o vetor de coeficientes da função objetivo primal, x é o vetor de variáveis de decisão primal, A é a matriz de coeficientes das restrições e b é o vetor de termos constantes das restrições.

A formulação dual do problema é definida por $max : u^T b$, sujeito a $u^T A \leq c$, onde u é o vetor de variáveis de decisão dual.

$$\begin{aligned} \max \quad & u^T b \\ \text{s.a.} \quad & u^T A \leq c \\ & u \text{ livre} \end{aligned}$$

Considere que estamos com uma bfs u do dual, seguimos os seguintes passos:

1. *Teste de otimalidade.* Se $B^{-1}b \geq 0$ então estamos na solução ótima. Caso contrário, encontre $k*$ tal que $(B^{-1}b)_{k*} < 0$.

2. *Direção de movimento.* Defina $d = (-I_{k*}.B^{-1}$.

3. *Computando a quantidade de movimento.* Se $\mathbf{d}N \leq \mathbf{0}$, então o dual é ilimitado. Caso contrário, defina t, e escolha $j*$, tal que

$$t* = \min \left\{ \frac{(C_n - \mathbf{u}N)_j}{(\mathbf{d}N)_j} : \quad (\mathbf{d}N)_j > 0 \right\} = \frac{(C_n - \mathbf{u}N)_{j*}}{(\mathbf{d}N)_{j*}}$$

4. Movendo na direção escolhida. Defina a nova solução dual básica viável u' tal que:

   - $u' = t * d$
   - B' é obtida a partir de B trocando a coluna k* de B pela coluna j* de N.

3.2 Dificuldades e descobertas na implementação

Este trabalho exigiu bastante manipulações das estruturas do Python: listas, dicionários e data-frames. Duas bibliotecas foram muito utilizadas para operações matemáticas com matrizes: numpy e pandas. Dentre as dificuldades enfrentadas, encontra-se a escassa quantidade de arquivos padronizados para efetuar testes do código.

4 Funcionamento básico e compilação do código

O arquivo disponibilizado tem extensão .ipynb e pode ser executado em um jupyter notebook utilizando o ambiente Anaconda ou com o Google Colab (ambiente recomendado pelos autores, pois as bibliotecas já estão instaladas, o que torna necessária apenas a importação das mesmas), sendo necessário para isso o upload do arquivo no ambiente escolhido.

O usuário deve utilizar arquivos de texto com o seguinte padrão:

- Suponha que tenhamos uma problema com m restrições e n variáveis. O arquivo terá o seguinte formato:
- Na primeira deve haver inteiros positivos separados por um espaço simples. Por exemplo, "t m n", onde
  1. o primeiro valor se refere ao tipo do problema. Adotaremos 0 para minimização e 1 para maximização
  2. o segundo valor se refere a quantidade de restrições do nosso problema
  3. o terceiro valor se refere a quantidade de variáveis do nosso problema.
- A segunda linha será referente a função objetivo e terá m valores do tipo decimal separadas por um espaço simples. Cada um desses valores se refere ao coeficiente da variável de mesma posição. Ou seja, se seu problema possui variáveis $x_1, \cdots, x_n$ então a i-ésima entrada dessa linha se refere ao coeficiente da função objetiva referente a i-ésima variável
- A partir da segunda linha teremos uma sequência de m pares de linhas referente as restrições. A primeira linha de cada par conterá um valor inteiro -1, 0 ou 1 referente ao tipo de restrição. Adotaremos -1 para restrição de menor igual, 0 para restrição de igual e 1 para restrição de maior ou igual.
- A segunda linha de cada par conterá uma sequência de n+1 valores decimais separados por espaço simples, onde cada entrada se refere ao coeficiente da variável daquela restrição e o último valor se refere ao valor a direita da desigualdade da restrição.
- Por fim, o arquivo conterá uma sequência de n restrições referentes às restrições de sinais das variáveis do seu problema. Cada restrição de sinal será formada por dois valores, um inteiro referente ao tipo de restrição (-1 se menor ou igual e 1 se maior ou igual) e um decimal referente ao valor de limitação. Caso a variável seja livre, a restrição terá apenas o valor 0.

  Outras instruções necessárias estarão escritas no código.