Programação Linear:
Introdução aos algoritmos polinomiais

Paulo Feofiloff

14/7/2001

Um **poliedro** é o conjunto de vetores que satisfazem dadas restrições lineares (por exemplo, o conjunto de vetores x com 4 componentes que satisfazem as restrições $11x_1 + 12x_2 + 13x_3 + 14x_4 \leq 15$, $21x_1 + 22x_2 + 23x_3 + 24x_4 = 25$, $31x_1 + 32x_2 + 33x_3 + 34x_4 \geq 35$, $x_1 \leq 0$ e $x_3 \geq 0$). Um **problema de viabilidade** consiste em determinar um vetor que pertença ao poliedro, isto é, um vetor que satisfaça todas as restrições (ou descobrir que tal vetor não existe). Um **problema de programação linear** consiste em encontrar um vetor em um poliedro dado que otimize uma dada função linear (por exemplo, minimize $91x_1 + 92x_2 + 93x_3 + 94x_4$).

O algoritmo Simplex resolve qualquer problema de programação linear[1] e portanto também qualquer problema de viabilidade. Por que, então, precisamos de outro algoritmo?

Exponencial versus polinomial

O Simplex consome muito tempo: ele pode executar até $2^{m/2}$ iterações antes de encontrar uma solução de um problema com m variáveis, ou seja, um problema em que os vetores têm m componentes.[2] Como m faz o papel de expoente na expressão que mede o número de iterações, dizemos que o Simplex é um algoritmo **exponencial**.

Para medir o consumo de tempo do algoritmo é melhor trocar o número de iterações pelo número de **operações aritméticas** (somas, subtrações,

[1] Na verdade, o Simplex resolve problemas canônicos primais. Mas qualquer problema de programação linear pode ser reduzido a um problema canônico primal equivalente.

[2] Para ficar consistente com a notação do livro, estou denotando o número de variáveis por m e não por n, como é tradicional.

divisões e multiplicações): se n é o número de restrições, o número total de operações aritméticas pode chegar a $2^{m/2}mn$ no pior caso, pois cada iteração executa cerca de mn operações aritméticas.

É bem verdade que esse número elevado de operações aritméticas só acontece quando os dados são particularmente desfavoráveis (veja seção 5.6 do livro), mas isso não nos isenta da responsabilidade de procurar um algoritmo **polinomial**, ou seja, um algoritmo que execute, por exemplo, não mais que m^9n^9, ou apenas m^2n^2 operações ariméticas, mesmo no pior caso. Infelizmente, parece que um tal algoritmo não existe. Mas existem algoritmos que são polinomiais num sentido um pouco mais fraco.

Os algoritmos polinomiais de programação linear — em particular o algoritmo do elipsóide e o de Yamnitsky-Levin — são radicalmente diferentes do Simplex. Por isso será necessário fazer uma longa discussão preliminar antes que possamos descrever um desses algoritmos. Faremos a discussão tendo em vista o algoritmo de Yamnitsky-Levin, mas muitas das observações preliminares aplicam-se igualmente bem ao algoritmo do elipsóide, muito mais famoso.

Discussão preliminar

Pior caso. O algoritmo de Yamnitsky-Levin não é sempre mais rápido que o Simplex, mas vence o Simplex nos casos em que o parâmetro m é grande e os dados são desfavoráveis ao Simplex.

Número de operações aritméticas versus tamanho dos coeficientes. O número de operações aritméticas que o Simplex executa depende apenas do número de variáveis e do número de restrições que definem o poliedro. O número de operações aritméticas não depende do valor dos coeficientes das restrições e da função objetivo.

No algoritmo de Yamnitsky-Levin, ao contrário, *o número de operações aritméticas depende do número total de dígitos dos coeficientes.*

Por exemplo, se o poliedro é definido por 10 restrições e 20 variáveis, o Simplex executa (em princípio, pelo menos) o mesmo número de operações aritméticas quer cada coeficiente tenha apenas 2 dígitos decimais quer tenha 200 dígitos decimais. Já no algoritmo de Yamnitsky-Levin, o número de operações aritméticas será 100 vezes maior no segundo caso. (Que estranho, não?)

Problemas de viabilidade. O algoritmo de Yamnitsky-Levin tem como missão resolver problemas de viabilidade e não problemas de programação linear. Assim, o algoritmo de Yamnitsky-Levin pode parecer mais "fraco" que o Simplex. Mas isso são apenas aparências, pois qualquer algoritmo que resolva problemas de viabilidade pode ser usado para resolver problemas de otimização.

Eis um pequeno exemplo. Suponha que queremos encontrar y que mazimize yb sob as restrições $yA \leq c$; basta então resolver o seguinte problema de viabilidade: encontrar vetores y e x que satisfaçam as restrições

$$yA \leq c, \quad Ax = b, \quad x \geq 0 \quad \text{e} \quad cx \leq yb\,. \tag{1}$$

O teorema da dualidade garante que y maximiza yb se e só se o par (y, x) satisfaz as restrições (1). (Veja mais na seção 14.2 do livro.)

Problema canônico dual. O algoritmo de Yamnitsky-Levin trata, especificamente, do problema da viabilidade para o poliedro canônico dual, ou seja, procura encontrar um vetor y que satisfaça as restrições

$$yA \;\leq\; c\,. \tag{2}$$

Mas é fácil verificar que o problema de viabilidade para qualquer poliedro pode ser reduzido a um poliedro canônico dual equivalente.

No que segue, denotaremos por n o número de colunas e m o número de linhas de A.

Dados inteiros. O algoritmo Simplex aceita quaisquer dados racionais. Para o algoritmo de Yamnitsky-Levin, entretanto, é conveniente supor que todos os dados são inteiros, ou seja, que todos os componentes de A e c em (2) são números inteiros. Se os dados são racionais mas não inteiros, basta multiplicar tudo por um número inteiro suficientemente grande para que tudo se torne inteiro.

O número ω. Precisamos ter uma boa delimitação superior dos valores absolutos dos componentes de todos os vetores que estão no poliedro. Em outras palavras, precisamos de um número ω tal que

$$\text{se } yA \leq c \text{ então } |y[i]| < \omega \text{ para todo } i. \tag{3}$$

Todos nós conhecemos a regra de Cramer (embora nem todos saibam por que ela vale): num sistema de equações lineares $Dx = d$, cada componente de qualquer solução x tem a forma α/δ, onde α e δ são determinantes de submatrizes da justaposição de D e d. (Veja os capítulos 10, 11 e 12 do livro.) Segue daí que se nosso poliedro $\{y : yA \leq c\}$ for limitado[3] então (3) vale com

$$\omega := \prod_i \prod_j (1 + \alpha_{ij}) \, ,$$

onde α_{ij} é o valor absoluto do componente (i, j) da justaposição de A e c.

E se o poliedro não for limitado? Nesse caso, só podemos dizer algo mais fraco: toda face minimal do poliedro tem um vetor y tal que $|y[i]| < \omega$ para todo i. Em particular, se as faces minimais são vértices então todo vértice tem components limitadas por ω. (Veja apêndices C e D do livro.) Felizmente, o algoritmo de Yamnitsky-Levin fica satisfeito com essa delimitação mais fraca se o poliedro não for limitado.

O número ω pode ser chamado de **tamanho**[4] do sistema (A, c). É evidente que ω é, em geral, um número enorme, certamente maior que mn. O número de dígitos de ω é essencialmente igual à soma dos números de dígitos de todos os componentes de A e c, pois $\log \omega = \sum_i \sum_j \log(1 + \alpha_{ij})$.

Volume de poliedro. O algoritmo de Yamnitsky-Levin depende, pelo menos informalmente, do conceito de "volume"[5] de um poliedro. Felizmente, basta definir o conceito de volume apenas para o tipo mais simples de poliedro: o tetraedro[6]. (Veja seção 15.2 do livro.)

O algoritmo de Yamnitsky-Levin não é capaz de decidir se o poliedro $\{y : yA \leq c\}$ é vazio, como sugerimos incorretamente acima. O algoritmo só é capaz de decidir se o poliedro tem "volume" não-nulo.[7] Para decidir se o poliedro é ou não vazio, é preciso submeter ao algoritmo um poliedro ligeiramente "inchado":

$$\{y : yA \leq c + c'\} \, ,$$

onde c' é o vetor cada um de cujos componentes vale $1/n\omega$. O truque de aumentar ligeiramente o poliedro original não é óbvio mas é muito interessante. A dificuldade está em determinar o valor de c' de modo que o novo

[3] Um poliedro é **limitado** se existe um número ξ tal que $|y[i]| < \xi$ para todo vetor y no poliedro e todo índice i.

[4] A rigor, eu deveria dizer que $\log \omega$ é o tamanho do sistema.

[5] Entre aspas porque estou usando a palavra em sentido apenas intuitivo, sem uma definição formal.

[6] Um tetraedro é o conjunto de todas as combinações convexas de $m + 1$ vetores.

[7] Por exemplo, uma folha de papel no espaço de três dimensões ($m = 3$) não é vazia mas tem volume nulo.

poliedro tenha "volume" não-nulo se e só se o poliedro original não é vazio. (Veja seção 14.4 do livro, que depende fundamentalmente da versão 13.2 do teorema da dualidade.)

Pontos interiores. Para contornar o complexo conceito de volume, o algoritmo de Yamnitsky-Levin trabalha com o conceito de ponto interior: um **ponto interior** de $\{y : yA \leq c\}$ é qualquer vetor z que satisfaça

$$zA[\,,j] \;<\; c[j]$$

para cada índice j. Informamente, o poliedro tem um ponto interior se e só se tem "volume" não-nulo.

O algoritmo de Yamnitsky-Levin

Agora podemos dizer precisamente (sem as meias-verdades que cometemos acima) o que o algoritmo de Yamnitsky-Levin faz: *a receber uma matriz inteira A e um vetor inteiro c, o algoritmo devolve um ponto interior do poliedro $\{y : yA \leq c\}$ ou informa que o poliedro não tem ponto interior.*

No início de cada iteração, temos um tetraedro que contém o poliedro $\{y : yA \leq c\}$ ou pelo menos uma parte relevante dele.[8] Cada iteração consiste no seguinte: o algoritmo calcula o centro, digamos t, do tetraedro; se t é um ponto interior de nosso poliedro então o algoritmo devolve t e pára; senão, calcula um novo tetraedro, de volume um pouco menor, que ainda contenha a parte relevante do poliedro dado. (Veja descrição e análise detalhadas no capítulo 15 do livro.)

Se o volume do tetraedro se tornar menor que $1/(4^m\omega^{2m})$, o algoritmo pára e anuncia que o poliedro dado não tem ponto interior. Isso está correto pois se o poliedro tivesse um ponto interior ele incluiria um pequeno tetraedro[9] de volume não inferior a $1/(4^m\omega^{2m})$. (Veja teorema 15.3 no livro.)

O algoritmo de Yamnitsky-Levin executa essencialmente

$$m^4(m+n)\log\omega$$

operações aritméticas, todas sobre números racionais.

Quem sabe uma boa maneira de começar a estudar o algoritmo de Yamnitsky-Levin é examinar o caso trivial em que a matriz A tem uma só linha.

[8] Se o poliedro é limitado, o tetraedro contém o poliedro todo.
[9] Esse pequeno tetraedro interno nada tem a ver com o grande tetraedro externo.

(Veja seção 15.12 do livro.) O caso em que A tem duas linhas também e muito instrutivo (veja exercício 15.6 do livro).