Minimização com Restrições de Igualdade
Método de penalização

Lucas Magno

1 Introdução

Lorem ipsum

2 Otimização Restrita

Diferentemente da otimização irrestrita, neste trabalho consideraremos problemas de otimização nos quais a região viável não é todo o $\mathbb{R}^n$ e sim um subconjunto seu determinado por várias equações não-lineares.

Podemos formular o problema na seguinte forma:

$$\begin{aligned} &\text{minimizar} && f(x) \\ &\text{sujeita a} && c_i(x) = 0, \quad i = 1, m \end{aligned} \tag{2.1}$$

onde $x \in \mathbb{R}^n$ e [1]

$$\begin{aligned} f: &\quad \mathbb{R}^n \to \mathbb{R} \\ c_i: &\quad \mathbb{R}^n \to \mathbb{R} \quad , \quad i = 1, m \end{aligned}$$

Portanto, não podemos mais utilizar os algoritmos já desenvolvidos e precisamos definir as condições de otimalidade neste novo problema. Para isto, é preciso que se defina uma condição de qualificação, isto é, hipóteses adicionais que nos permitirão qualificar os minimizadores.

[1] Assumindo f e c_i continuamente diferenciáveis.

2.1 LICQ e Multiplicadores de Lagrange

Uma qualificação possível (de fato a mais fraca [Wachsmuth, 2013]) é a *Linear Independence Constraint Qualification* ou LICQ, definida a seguir.

Definição 2.1. *Um ponto x^* é dito satisfazer LICQ se e somente o conjunto formado pelos gradientes das restrições em x^**

$$\{\nabla c_1(x^*), \ldots, \nabla c_m(x^*)\}$$

é linearmente independente.

Assim, podemos agora enunciar a condição necessária de primeira ordem [Friedlander, 1994]:

Teorema 2.1. *Seja x^* um minimizador local do problema 2.1 que satisfaça LICQ, então existe $\lambda^* \in \mathbb{R}^m$ tal que*

$$\begin{aligned} \nabla \mathcal{L}(x^*) &= \nabla f(x^*) - \sum_{i=0}^{m} \lambda_i^* \nabla c_i(x^*) \\ &= 0 \end{aligned} \tag{2.2}$$

onde λ^* é conhecido como o vetor de multiplicadores de langrange.

Uma vez que temos uma condição de otimalidade, podemos a utilizar como critério de parada no algoritmo a ser implementado.

Vale notar também que podemos generalizar este método para restrições de *desigualdade*, da forma

$$c_i(x) \geq 0$$

e então obtemos as condições de Karush-Kuhn-Tucker (KKT). [Nocedal and Wright, 2006]

3 Método de Penalização

Mas como lidar com as restrições? A solução simples é não lidar com elas! [Friedlander, 1994] Quer dizer, podemos definir o seguinte problema irrestrito

$$\underset{x \in \mathbb{R}^n}{\text{minimizar}} \quad Q(x, \mu) = f(x) + \frac{1}{2\mu} \sum_{i=0}^{m} c_i(x)^2 \tag{3.1}$$

para dado $\mu \in \mathbb{R}$, no qual pontos fora da região viável recebem uma *penalidade* com peso $1/2\mu$ e assim, para μ suficientemente grande, espera-se que o algoritmo descarte pontos fora da região viável.

De fato, pode-se mostrar que, sendo a sequência $\{\mu_k\}_{k=0}^{\infty}$ tal que

$$\mu_k \to 0 \quad (k \to \infty) \tag{3.2}$$

então a sequência $\{x_k\}_{k=0}^{\infty}$ de soluções dos subproblemas

$$\underset{x \in \mathbb{R}^n}{\text{minimizar}} \quad Q(x, \mu_k)$$

tende a x^*, que é solução do problema 2.1.

No entanto, isso requer a solução exata do subproblema 3.1. Por outro lado, se tivermos x_k tal que

$$\|\nabla Q(x_k, \mu_k)\| \leq r_k, \quad r_k \in \mathbb{R}$$

e

$$r_k \to 0$$

também é possível mostrar que $x_k \to x^*$ e

$$\lambda_i^k = -\frac{c_i(x_k)}{\mu_k} \to \lambda_i^* \quad i = 1, m$$

o que é muito mais útil de um ponto de vista prático.

4 Algoritmo

Tendo isto, podemos montar um algoritmo a fim de resolver o problema 2.1.

Dados $\{\mu_k\}_{k=0}^{\infty}$ e $\{r_k\}_{k=0}^{\infty}$ como definidos acima e $x_0^S \in \mathbb{R}^n$

Algorithm 4.1 Método de Penalização

for $k = 0, 1, 2, \ldots$ **do**

Passo 1 Usando x_k^S como ponto inicial, encontrar x_k solução aproximada do subproblema 3.1 tal que $\|\nabla Q(x_k, \mu_k)\| \leq r_k$

Passo 2 Se x_k satisfaz o critério 2.2, parar.

Passo 3 Escolher x_{k+1}^S

Passo 4 Escolher μ_{k+1}

end for

Por motivo de simplicidade, serão usadas as fórmulas

$$r_k = \epsilon$$

$$x_{k+1}^S \leftarrow x_k$$

$$\mu_{k+1} \leftarrow \mu_k / 2$$

onde ϵ é a tolerância definida para o programa.

Além disso, definimos " satisfazer o critério 2.2" como

$$\|c(x)\| \leq \epsilon \quad \text{(viabilidade)}$$
$$\|\nabla \mathcal{L}(x)\| \leq \epsilon \quad \text{(otimalidade)}$$

4.1 Mal condicionamento

Como método no primeiro passo foi escolhido o de Newton, no entanto é necessário uma observação. Para encontrar uma direção de descida d, o método resolve um sistema na forma

$$\nabla^2 Q(x,\mu)d = -\nabla Q(x,\mu)$$

Porém, como

$$\begin{aligned}\nabla^2 Q(x,\mu) &= \nabla^2 f(x) + \frac{1}{\mu}\sum_{i=0}^{m} c_i(x)\nabla^2 c_i(x) + \frac{1}{\mu}\sum_{i=0}^{m}\nabla c_i(x)\nabla c_i(x)^\top \\ &= \nabla^2 \mathcal{L}(x) + \frac{1}{\mu}\sum_{i=0}^{m}\nabla c_i(x)\nabla c_i(x)^\top\end{aligned} \tag{4.1}$$

nota-se que o último termo é uma matriz de posto m e, para μ suficientemente pequeno, vale que $\nabla^2 Q(x,\mu)$ é mal condicionada, o que pode inviabilizar a solução do sistema.

Em seu lugar, podemos utilizar um sistema equivalente introduzindo uma variável auxiliar $z \in \mathbb{R}^m$:

$$\begin{bmatrix}\nabla^2 \mathcal{L}(x) & A(x)^\top \\ A(x) & -\mu I\end{bmatrix}\begin{bmatrix}d \\ z\end{bmatrix} = \begin{bmatrix}-\nabla Q(x,\mu) \\ 0\end{bmatrix}$$

onde

$$A(x) = \begin{bmatrix}\nabla c_1(x) \\ \vdots \\ \nabla c_m(x)\end{bmatrix}$$

Esse sistema não apresenta mal condicionamento devido a μ pequeno e portanto será utilizado na implementação do método de Newton, mas fora isso o método se mantém o mesmo do trabalho anterior.

5 O Programa

Para implementação do algoritmo foi feito um programa em `FORTRAN 2008`, usando como base o código já desenvolvido anteriormente, portanto aqui só serão mencionadas as adições referentes a este trabalho.

5.1 Módulos

Os módulos novos criados foram:

`constrained` Implementa o método de penalização bem como o de Newton modificado.

`stats2` Fornece *wrappers* e subrotinas para automatizar a contagem e a impressão de diversas etapas durante a execução dos algoritmos.

`test` Implementa vários problemas de teste para o programa.

5.2 Arquivos

Os arquivos mantiveram a hierarquia, mas foi criado um diretório `data/` para guardar os resultados da execução (`.dat`) e também os códigos para geração dos gráficos (`.gp`), através do programa `gnuplot`.

5.3 Compilação

A compilação manteve a mesma estrutura:

`make EP2` Compila o programa, criando o executável.

`make tex2` Gera os gráficos e compila o relatório, criando o arquivo `.pdf`.

`make clean` Realiza a limpeza, deletando os arquivos de saída.

A única observação é que, para o relatório, foi utilizado o programa `rubber` [Kishimoto, 2009], em sua versão 1.4.

6 Resultados

Foram escolhidos quatro problemas para testar o programa, definidos no $\mathbb{R}^2$ a fim de ilustração, cujos resultados foram:

	$\|c^*\|$	$\|\nabla\mathcal{L}^*\|$	sub	f	∇f	$\nabla^2 f$	armijo	norma	ângulo
teste 1	0.60E-06	0.50E-12	25	2368	1000	949	470	0	5320
teste 2	0.98E-06	0.37E-08	36	7934	3979	3906	122	0	16251
teste 3	0.60E-06	0.91E-06	26	2816	1099	1046	724	0	5774
teste 4	FC								

Tabela 6.1: Saída do programa para os quatro problemas testados, com $\epsilon = 10^{-6}$, $\theta = 10^{-5}$, $\gamma = \sigma = 10^{-4}$ e ponto inicial $(1, 1)$.

E a seguir vamos analisar cada um.

6.1 Não-Linearidade das restrições

Os problemas 1 e 2 são, respectivamente

$$\begin{aligned} &\text{minimizar} && x^2 + y^2 \\ &\text{sujeita a} && x + y - 1 = 0 \end{aligned} \qquad\qquad \begin{aligned} &\text{minimizar} && x^2 + y^2 \\ &\text{sujeita a} && (x + y - 1)^3 = 0 \end{aligned}$$

Ou seja, ambos consistem em achar o ponto da reta $x + y = 1$ mais próximo à origem e a única diferença entre eles é que a restrição é linear no primeiro, mas não no segundo.

Visualizando o progresso do algoritmo através dos gráficos

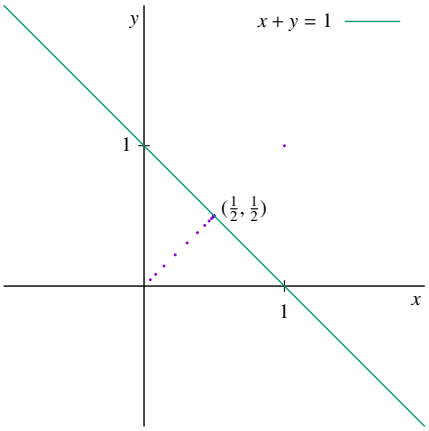

Figura 6.1: Sequência de soluções x_k para o problema 1.

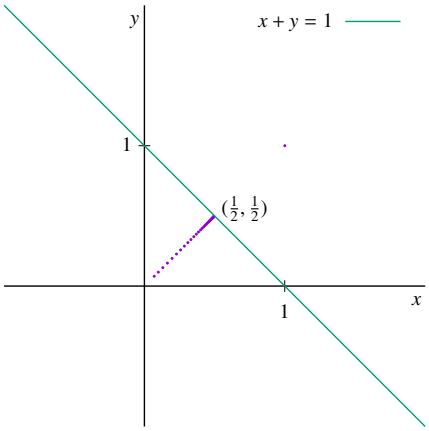

Figura 6.2: Sequência de soluções x_k para o problema 2.

fica claro que ambos chegam à mesma solução de forma similar. No entanto, tanto pelo gráfico quanto pela tabela de resultados 6, nota-se que o segundo caso resolve um número maior de subproblemas e ainda assim termina com uma precisão pior do que o primeiro, então restrições não-lineares introduzem uma certa dificuldade comparadas às lineares para este algoritmo.

6.2 LICQ

Para ilustrar a necessidade da LICQ para o método, dois outros problemas foram testados:

minimizar	$x^2 + y^2$	minimizar	$x^2 + y^2$
sujeita a	$x = 1$	sujeita a	$(x-1)^2 + (y-1)^2 = 1$
	$y = 0$		$(x-1)^2 + (y+1)^2 = 1$

nos quais em ambos a região viável se restringe a um único ponto, $x^* = (1, 0)$, e cujos gradientes da função e das restrições nesse ponto são, respectivamente:

$$\nabla f(x^*) = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \quad \nabla c_1(x^*) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \nabla c_2(x^*) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\nabla f(x^*) = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \quad \nabla c_1(x^*) = \begin{bmatrix} 0 \\ -2 \end{bmatrix}, \quad \nabla c_2(x^*) = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

em que fica claro que, enquanto os gradientes das restrições do primeiro são linearmente independente, os do segundo não o são.

Novamente, podemos observar os gráficos:

e se nota que ambos se aproximam da solução x^*. Apesar disso, como visto na tabela 6, o programa não converge para o problema 4.

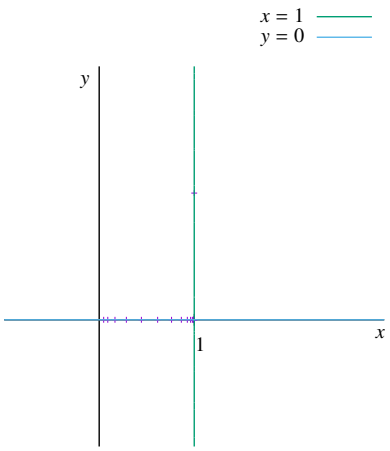

Figura 6.3: Sequência de soluções x_k para o problema 3.

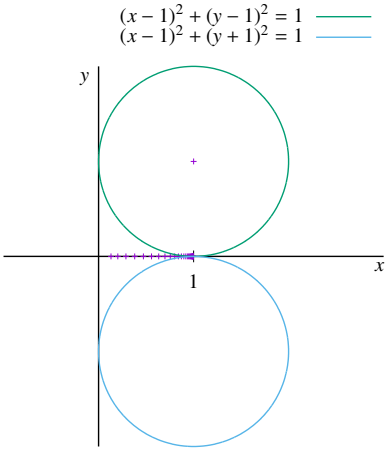

Figura 6.4: Sequência de soluções x_k para o problema 4.

A partir da evolução dos valores dos critérios de parada para os últimos pontos do algoritmo em ambos os problemas

$\|c(x)\|$	$\|\nabla\mathcal{L}(x)\|$
0.95E-05	0.54E-06
0.48E-05	0.11E-11
0.24E-05	0.11E-10
0.12E-05	0.28E-11
0.60E-06	0.91E-06

Tabela 6.2: Valores dos critérios de parada para a sequência x_k do problema 3.

$\|c(x)\|$	$\|\nabla\mathcal{L}(x)\|$
0.10E-03	0.11E-07
0.63E-04	0.20E-06
0.40E-04	0.31E-06
0.25E-04	0.53E-06
0.16E-04	0.32E-07

Tabela 6.3: Valores dos critérios de parada para a sequência x_k do problema 4.

somado ao fato de que o programa resolveu 26 subproblemas para o problema 3 e 28 para o 4 (embora não esteja informado na tabela 6), observa-se que a convergência se dá muito mais rapidamente para o primeiro, e é possível ver o ponto em que atinge a região viável, na qual vale a condição de parada (dentro da tolerância).

Para o segundo problema, porém, o algoritmo acaba não alcançando a região viável, pois, conforme ele se aproxima dela, os gradientes das restrições se tornam linearmente dependentes, de forma que não seja possível anular o gradiente da Lagrangeana.

7 Conclusão

Referências

A. Friedlander. *Elementos de Programação Não-Linear*. Editora UNICAMP, 1 edition, 1994.

P. N. Kishimoto. Rubber in launchpad, November 2009. URL `https://launchpad.net/rubber/`.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

G. Wachsmuth. On {LICQ} and the uniqueness of lagrange multipliers. *Operations Research Letters*, 41(1):78 – 80, 2013. ISSN 0167-6377. doi: http://dx.doi.org/10.1016/j.orl.2012.11.009. URL `http://www.sciencedirect.com/science/article/pii/S0167637712001459`.