O Método da Potência para autovalor e autovetor dominantes

Prof. Doherty Andrade

www.metodosnumericos.com.br

1 Introdução ao método da potência (power)

O método da potência é um método iterativo que calcula aproximações para o maior autovalor em módulo e seu autovetoer correspondente. Embora, seja limitado em aplicações, tem grande utilidade.

Seja A uma matriz quadrada de ordem n. Difinimos o quociente de Rayleigh de um vetor não nulo $x \in R^n$ por

$$r(x) = \frac{x^T A x}{x^T x}$$

.

Note que se x é um autovetor de A associado ao autovalor λ, então

$$r(x) = \frac{x^T A x}{x^T x} = \frac{\lambda x^x}{x^T x} = \lambda$$

que é exatamente o autovalor correspondente.

O método da potência se baseia na existência de n autovetores de A. Se este é o caso, a matriz A é diagonalizável.

Sejam $u_1, u_2, \ldots, u_n$ autovetores de A associados, respectivamente, aos autovalores $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n| > 0$. Note que

$$A^k u_j = \lambda_j^k u_j,$$

para todo autovetor de A.

Ao maior autovalor em módulo chamamos de autovalor dominante e o autovetor correspondente de autovetor dominante.

Consideremos o vetor $v^{(0)} = c_1 u_1 + c_2 u_2 + \cdots + c_n u_n$\$ com $c_1 \neq 0$.

Segue que

$$Av^{(0)} = c_1 \lambda_1 u_1 + c_2 \lambda_2 u_2 \cdots + c_n \lambda_n u_n.$$

E assim por diante,

$$A^k v^{(0)} = c_1 \lambda_1^k u_1 + c_2 \lambda_2^k u_2 \cdots + c_n \lambda_n^k u_n.$$

Ou equivalentemente,

$$A^k v^{(0)} = \lambda_1^k \left(c_1 u_1 + c_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k u_2 \cdots + c_n \left(\frac{\lambda_n}{\lambda_1} \right)^k u_n \right).$$

Como λ_1 é o maior autovalor de A, segue que

$$\lim_{k\to\infty}\left(\frac{\lambda_i}{\lambda_1}\right)^k = 0$$

para cada $i = 1, 2, \ldots, n$. Portanto,

$$\lim_{k\to\infty} A^k v^{(0)} = \lim_{k\to\infty} c_1 \lambda_1^k u_1.$$

Segue que uma aproximação para u_1 é

$$u_1 \approx \frac{A^k v^{(0)}}{\|A^k v^{(0)}\|}.$$

O método da potência determina uma aproximação para o autovetor dominante.
O método da potência pode ser resumindo assim:

(1) Tome um vetor inicial $v^{(0)}$ tal que $\|v^{(0)}\| = 1$

(2) Para $k = 1, 2, \ldots$

(3) Faça $w = Av^{(k-1)}$

(4) Faça $v^{(k)} = \frac{w}{\|w\|}$.

A cada iteração, a sequência $v^{(k)}$ converge para o autovetor u_1 dominante.
Uma vez tendo obtido o autovetor dominante u_1, segue que

$$\lambda_1 = \frac{u_1^T A u_1}{u_1^T u_1}.$$

2 Algoritmo em Python e exemplo

```
In [8]: import numpy as np
        import matplotlib.pyplot as plt
        import scipy.linalg as la
```

```
In [9]: # metodo da potência para determinar
        # autovalor e autovetoer dominantes

        def powervalor(A, v):
            Av = A.dot(v)
            return v.dot(Av)

        def power_iteracao(A):
            n, d = A.shape

            v = np.ones(d) / np.sqrt(d)
            ev = powervalor(A, v)
```

```
    while True:
        Av = A.dot(v)
        v_new = Av / np.linalg.norm(Av)

        ev_new = powervalor(A, v_new)
        if np.abs(ev - ev_new) < 0.0001:
            break

        v = v_new
        ev = ev_new
    return ev_new, v_new
#Exemplo
# A = np.array([[4,5,6],[6,5,0],[1,2,-1]])
# power_iteracao(A)
```

Vejamos um exemplo: determinar o autovalor e autovetor dominantes da matriz

$$A = \begin{bmatrix} 4 & 5 & 6 \\ 6 & 5 & 0 \\ 1 & 2 & -1 \end{bmatrix}$$

```
In [10]: A = np.array([[4,5,6],[6,5,0],[1,2,-1]])
         A
```

```
Out[10]: array([[ 4,  5,  6],
                [ 6,  5,  0],
                [ 1,  2, -1]])
```

```
In [11]: power_iteracao(A)
```

```
Out[11]: (10.769696956831412, array([0.68205475, 0.70918762, 0.17847757]))
```

Vamos confirmar usando o pacote de álgebra linear do Python.

```
In [12]: autos = la.eig(A)
         print(autos[0])
         print(autos[1]) #autovetor deve ser lido em coluna
```

```
[10.7696659 +0.j         -1.38483295+1.70605218j -1.38483295-1.70605218j]
[[ 0.6820084 +0.j          0.69303571+0.j          0.69303571-0.j        ]
 [ 0.70923524+0.j         -0.60786414-0.16242366j -0.60786414+0.16242366j]
 [ 0.17846546+0.j         -0.1154268 +0.33241223j -0.1154268 -0.33241223j]]
```

Confere!

```
In [ ]:
```