# Algebra Linear Computacional - Lista de Exercicios 2

Lucas Resende Pellegrinelli Machado (2018126673)

March 26, 2019

**Exercício 1.**
Temos a matriz:

$$M = \begin{bmatrix} 3 & a_0 & a_1 \\ a_2 & 4 & a_3 \\ a_4 & 12 & 42 \\ a_5 & a_6 & 28 \\ 1 & 2 & a_7 \end{bmatrix}$$

E sabemos que ela tem posto 1. Logo uma das linhas é linearmente independente e as outras são linearmente dependentes.

Para resolver o problema, podemos ver a linha

$$M_4 = \begin{bmatrix} 1 & 2 & a_7 \end{bmatrix}$$

E concluir que dado $Ci$ uma coluna da matriz, temos que $2C_1 = C_0$

Da mesma forma, podemos observar na linha

$$M_2 = \begin{bmatrix} a_4 & 12 & 42 \end{bmatrix}$$

E concluir que $\frac{7}{2}C_2 = C_1$

Assim conseguimos completar todos os outros valores da matriz

$$M = \begin{bmatrix} 3 & 6 & 21 \\ 2 & 4 & 14 \\ 6 & 12 & 42 \\ 4 & 8 & 28 \\ 1 & 2 & 7 \end{bmatrix}$$

Essa matriz possui possui posto 1 como o enunciado pedia visto que todas as linhas são multiplas da linha

$$M_4 = \begin{bmatrix} 1 & 2 & 7 \end{bmatrix}$$

**Exercício 2.**

.

a. Utlizando do vetor de média das avaliações dos filmes:

$$m = \begin{bmatrix} 1.8 & 3.0 & 4.0 & 2.4 & \dots \end{bmatrix}$$

Temos que

$$B = \begin{bmatrix} 3-1.8 & 4-3.0 & 3-4.0 & 1-2.4 & \dots \\ 1-1.8 & 2-3.0 & 5-4.0 & 3-2.4 & \dots \\ 2-1.8 & a_0-3.0 & a_1-4.0 & 4-2.4 & \dots \\ 1-1.8 & a_2-3.0 & a_3-4.0 & 1-2.4 & \dots \\ 2-1.8 & 4-3.0 & 5-4.0 & 3-2.4 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} 1.2 & 1.0 & -1.0 & -1.4 & \dots \\ -0.8 & -1.0 & 1.0 & 0.6 & \dots \\ 0.2 & a_0-3.0 & a_1-4.0 & 1.6 & \dots \\ -0.8 & a_2-3.0 & a_3-4.0 & -1.4 & \dots \\ 0.2 & 1.0 & 1.0 & 0.6 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

b. Usando o vetor de média das notas dadas por usuário

$$u = \begin{bmatrix} -0.05 \\ -0.05 \\ 0.45 \\ -1.05 \\ 0.7 \end{bmatrix}$$

Temos que

$$C = \begin{bmatrix} 1.2+0.05 & 1.0+0.05 & -1.0+0.05 & -1.4+0.05 & \dots \\ -0.8+0.05 & -1.0+0.05 & 1.0+0.05 & 0.6+0.05 & \dots \\ 0.2-0.45 & (a_0-3.0)-0.45 & (a_1-4.0)-0.45 & 1.6-0.45 & \dots \\ -0.8+1.05 & (a_2-3.0)+1.05 & (a_3-4.0)+1.05 & -1.4+1.05 & \dots \\ 0.2-0.7 & 1.0-0.7 & 1.0-0.7 & 0.6-0.7 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} =$$

$$= \begin{bmatrix} 1.25 & 1.05 & -0.95 & -1.45 & \dots \\ -0.75 & -0.95 & 1.05 & 0.65 & \dots \\ -0.25 & a_0-3.45 & a_1-4.45 & 1.15 & \dots \\ 0.25 & a_2-1.95 & a_3-2.95 & -0.35 & \dots \\ -0.5 & 0.3 & 0.3 & 0.1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

c. Temos que primeiro achar $C^*$ pelo produto de $U$, $\Sigma$ e $V^T$. Como deixamos $C$ em função de $a_0$, $a_1$, $a_2$ e $a_3$, depois de conseguir $C^*$, é só igualar as matrizes para achar os valores desejados.

$$C^* = U \times \Sigma \times V^T = \begin{bmatrix} 1.186 & 1.113 & -1.204 & -1.099 & \dots \\ -0.822 & -0.904 & 0.884 & 0.885 & \dots \\ -0.217 & -1.479 & 0.695 & 0.995 & \dots \\ 0.375 & 0.959 & -0.606 & -0.725 & \dots \\ -0.522 & 0.311 & 0.232 & -0.015 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Logo, igualando os valores correspondentes as posições de $a_0$, $a_1$, $a_2$ e $a_3$ na matriz $C^*$ com os valores da matriz $C^*$

$$\begin{cases} a_0 - 3.45 = -1.49 \\ a_1 - 4.45 = 0.695 \\ a_2 - 1.95 = 0.959 \\ a_3 - 2.95 = -0.606 \end{cases}$$

Assim temos que

$$\begin{cases} a_0 = 1.960 \approx 2 \text{ (nota do usuário 2 ao filme 1)} \\ a_1 = 5.145 \approx 5 \text{ (nota do usuário 2 ao filme 2)} \\ a_2 = 2.909 \approx 3 \text{ (nota do usuário 3 ao filme 1)} \\ a_3 = 2.344 \approx 2 \text{ (nota do usuário 3 ao filme 2)} \end{cases}$$

**Exercício 3.**

.

a. Falso. O NMF requer que as 3 matrizes $V$, $H$ e $W$ tenham todos os seus elementos não negativos, logo é impossível que $A$ possua elementos negativos e seja reconstruida de forma perfeita. Caso a matriz tivesse valores negativos, existe a opção de utilizar o Semi-NMF, que se assemelha com o NMF mas é menos restrito quanto ao sinal dos valores da matriz.

b. Verdadeiro. O SVD truncado gera a melhor aproximação de uma matriz para qualquer posto dado, logo o NMF tem que ser no máximo equivalente com o SVD truncado.

**Exercício 4.**
Dado a matriz

$$A = \begin{bmatrix} 3 & 0 & 2 \\ 9 & 1 & 7 \\ 1 & 0 & 1 \end{bmatrix}$$

a. Como a norma-1 de uma matriz $m \times n$ é definida por

$$||A||_1 = \max_{1 \le j \le n} \sum_{i=1}^{m} |a_{i,j}|$$

Ou seja, a soma dos valores absolutos da coluna cuja soma é a maior dentre todas as colunas.

Na matriz em questão temos (considerando $S_i$ como a soma dos valores absolutos da coluna $i$):

$S_0 = |3| + |9| + |1| = 13$

$S_1 = |0| + |1| + |0| = 1$

$S_2 = |2| + |7| + |1| = 10$

$$||A||_1 = \max\{S_0, S_1, S_2\} = S_0 = 13$$

b. Já a norma-infinito é definida por

$$||A||_\infty = \max_{1 \leq i \leq m} \sum_{i=1}^{n} |a_{i,j}|$$

Ou seja, é análoga à norma-1 porém com a soma das linhas ao invés das colunas. Na matriz em questão temos

$S_0 = |3| + |0| + |2| = 5$

$S_1 = |9| + |1| + |7| = 17$

$S_2 = |1| + |0| + |1| = 2$

$$||A||_1 = \max\{S_0, S_1, S_2\} = S_1 = 17$$

c. A norma-2 é dada por

$$||A||_2 = \sqrt{\delta_{max}(A^T \times A)}$$

Sendo $\delta_{max}(M)$ uma função que retorna o maior autovalor de uma matriz $M$.

Utilizando o Python e a biblioteca Numpy, podemos calcular a norma-2 da matriz da questão da forma:

```
import numpy as np

# Matriz do enunciado
A = np.array([[3, 0, 2],
              [9, 1, 7],
              [1, 0, 1]])

# Utilizando o segundo parametro da funcao 'norm' indicando que queremos a norma-2
norma_2 = np.linalg.norm(A, 2)
print(norma_2)
```

Que nos dá o resultado de

$$||A||_2 \approx 12.0748$$

d. A norma Frobenius por sua vez é dada por

$$||A||_F = \sqrt{\sum_{i,j} |a_{i,j}|^2}$$

Que na matriz em questão é

$$||A||_F = \sqrt{|3|^2 + |2|^2 + |9|^2 + |1|^2 + |7|^2 + |1|^2 + |1|^2} = \sqrt{146} \approx 12.083$$

**Exercício 5.**

.

a. Como o SVD de posto $k$, para representar uma matriz $C_{m\times n}$ precisa de 2 matrizes, $U_{k\times m}$ e $V^T_{n\times k}$ e um vetor $\Sigma_k$. Isso implica que precisamos armazenar $mk$ números na matriz $U$, $nk$ números na matriz $V^T$ e mais $k$ números no vetor $\Sigma$, logo o número de bytes a serem armazenados para o SVD é

$$n_{bytes} = (mk + nk + k) = k(m + n + 1) = 1793k$$

Para saber o valor máximo de $k$ para que o SVD valha a pena, precisamos ver qual o valor máximo de $k$ para que a quantidade de bytes a serem armazenados após os SVD seja menor que a quantidade de bytes da matriz original.

Como no exemplo da questão estamos trabalhando com uma matriz de $1024 \times 768$, temos um total de 786432 bytes na imagem original. Para achar o $k$ máximo, precisamos apenas resolver a inequação:

$$1793k \leq 786432 \implies k \leq 438$$

Assim o valor de $k$ tem que ser menor ou igual a 438 para que o SVD valha a pena no sentido de poupar espaço.

b. Com um SVD truncado de uma matriz $C_{m\times n}$, gastamos $mk$ bytes para os valores de $U_{m\times m}$, $nk$ bytes para os valores de $V^T_{n\times n}$ e k bytes para o vetor $\Sigma_{\min\{m,n\}}$.

Para 10 imagems, teremos uma matriz $C_{786432\times 10}$ (uma imagem por linha). Dessa forma teremos que gastar

$$mk + nk + k = 786432k + 10k + k = 786443k \text{ bytes}$$

Já para 1000 imagems, com uma matriz $C_{786432\times 1000}$ teremos que gastar

$$mk + nk + k = 786432k + 1000k + k = 787433k \text{ bytes}$$