BOOK: Linear Algebra: Theory, Intuition, Code

AUTHOR: Mike X Cohen

WEBSITE: sincxpress.com

CHAPTER: Rank (chapter 7)

In [14]:
```
## import libraries for the entire chapter
import numpy as np
```

Section 7.3, code block 7.1

RANGO di una MATRICE

In [15]:
```
A = np.zeros([3,3])
```

In [16]:
```
r = np.linalg.matrix_rank(A)
print(r)
```

```
0
```

Section 7.4, code block 7.3

SE MOLTIPLICO PER UNO SCALARE "*r*" NON CAMBIA

In [17]:
```
# scalar and matrix
s = np.random.randn()
M = np.random.randn(3,5)

# their ranks
r1 = np.linalg.matrix_rank(M)
r2 = np.linalg.matrix_rank(s*M)

# same or different?
print(r1,r2)
```

```
3 3
```

Section 7.5, Rank of added matrices

RANGO di MATRICI SOMMATE

In [18]:
```
A = np.zeros([3,3])
A[0,0] = 1
B = np.zeros([3,3])
B[1,1] = 1
C = A+B
```

```
print('Matrice A\n',A,'\n')
print('Matrice B\n',B,'\n')
print('Matrice C\n',C,'\n')
r1 = np.linalg.matrix_rank(A)
r2 = np.linalg.matrix_rank(B)
r3 = np.linalg.matrix_rank(C)
print('Rango di A = ',r1,'Rango di B = ',r2,'Rango di C = ',r3)
```

```
Matrice A
 [[1. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]

Matrice B
 [[0. 0. 0.]
 [0. 1. 0.]
 [0. 0. 0.]]

Matrice C
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 0.]]

Rango di A =  1 Rango di B =  1 Rango di C =  2
```

Section 7.6, Rank of multiplied matrices

RANGO di MATRICI MOLTIPLICATE

$$\begin{bmatrix}1&2&3\\3&4&1\\5&9&1\end{bmatrix}_3\begin{bmatrix}0&3&5\\1&0&4\\3&3&0\end{bmatrix}_3=\begin{bmatrix}11&12&13\\7&12&31\\12&18&61\end{bmatrix}_3 \quad C1 \qquad (7.9)$$

$$\begin{bmatrix}1&2&0\\3&4&0\\5&9&0\end{bmatrix}_2\begin{bmatrix}0&0&5\\0&0&4\\0&0&1\end{bmatrix}_1=\begin{bmatrix}0&0&13\\0&0&31\\0&0&61\end{bmatrix}_1 \quad C2 \qquad (7.10)$$

$$\begin{bmatrix}1&1&1\\2&2&2\\3&3&0\end{bmatrix}_2\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix}_0=\begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix}_0 \quad C3 \qquad (7.11)$$

$$\begin{bmatrix}-1&-4&2\\-4&2&-1\\9&4&-3\end{bmatrix}_3\begin{bmatrix}1&4&0\\4&-2&0\\-9&-4&0\end{bmatrix}_2=\begin{bmatrix}-35&-4&0\\13&-16&0\\52&40&0\end{bmatrix}_2 \quad C4 \qquad (7.12)$$

In [19]:
```
import numpy as np
C1 = [[11,12,13],[7,12,31],[12,18,61]]
C2 = [[0,0,13],[0,0,31],[0,0,61]]
C3 = [[0,0,0],[0,0,0],[0,0,0]]
C4 = [[-35,-4,0],[13,-16,0],[52,40,0]]
r1 = np.linalg.matrix_rank(C1)
r2 = np.linalg.matrix_rank(C2)
r3 = np.linalg.matrix_rank(C3)
```

```
r4 = np.linalg.matrix_rank(C4)
print('Rango di C1 =',r1,'Rango di C2 =',r2,'Rango di C3 =',r3,'Rango di C4 =',r
```

```
Rango di C1 = 3 Rango di C2 = 1 Rango di C3 = 0 Rango di C4 = 2
```

Section 7.7, Rank in case of transposition-multiplication

RANGO di MATRICI A, A^T, A^TA, and AA^T

Esse hanno tutte lo stesso Rango

In [10]:
```
# two random matrices
A = np.random.randn(9,2)
B = np.random.randn(2,16)
print('Rango di A = ',np.linalg.matrix_rank(A))
print('Rango di B = ',np.linalg.matrix_rank(B))

# their product (assume max possible rank)
C = A@B
print('Rango di A@B = ',np.linalg.matrix_rank(C))
```

```
Rango di A =  2
Rango di B =  2
Rango di A@B =  2
```

Section 7.8, Rank of random matrices

RANGO di MATRICI RANDOM

Qui la probabilità di avere righe-colonne linearmente dipendenti è infinitesima --> hanno il massimo Rango possibile

Le cose cambiano se i numeri sono sorteggiati in un campo ristretto

Per esempio se scelgo solo tra due numeri {0,1}

In [22]:
```
import numpy as np
C1 = [[0,0,0,0],[0,0,0,0],[1,0,1,1],[1,1,0,0]]
r1 = np.linalg.matrix_rank(C1)
print('Rango di C1 =',r1)
```

```
Rango di C1 = 2
```

Section 7.9, Full-rank matrices via “shifting”

FULL RANK nel caso di "SHIFTING"

$$\begin{bmatrix} 1 & 3 & -19 \\ 5 & -7 & 59 \\ -5 & 2 & -24 \end{bmatrix}_2 + .01 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_3 = \begin{bmatrix} 1.01 & 3 & -19 \\ 5 & -6.99 & 59 \\ -5 & 2 & -23.99 \end{bmatrix}_3 \tag{7.25}$$

Section 7.10, code block 7.5

DIIFICULTIES in COMPUTING RANK in PRACTICE

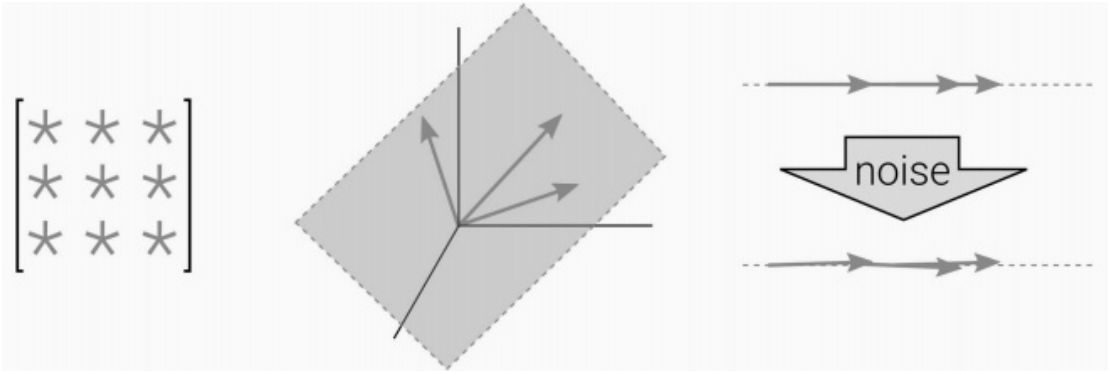
Vorrei spiegarti perché è difficile calcolare il rango di matrici di grandi dimensioni, sia da un'interpretazione algebrica che geometrica. Ho scritto sopra che un modo per calcolare il rango di una matrice è contare il numero di valori singolari diversi da zero. Non hai ancora imparato i valori singolari, ma per ora è sufficiente sapere che una matrice M × N ha min{M, N} valori singolari. I computer soffrono di errori di arrotondamento che portano a incertezze nel distinguere i numeri molto piccoli dallo zero reale. Spesso, i numeri inferiori a circa 10^{-15} sono considerati pari a zero più l'errore di arrotondamento del computer (l'esponente esatto dipende dalla precisione del computer).

Quindi diciamo che il computer stima che un valore singolare sia 3×10^{-15}; si tratta di un vero valore singolare diverso da zero che sembra essere davvero piccolo o in realtà è zero ma con errore di arrotondamento? Il tuo software per computer (MATLAB o Python) definirà una soglia per considerare un piccolo valore singolare uguale a zero. Ma quella soglia è arbitraria e un errore di arrotondamento potrebbe trovarsi casualmente su entrambi i lati di quel confine. Pertanto, l'errore di arrotondamento più una soglia arbitraria può influenzare il rango riportato della matrice. Torneremo su questo argomento, inclusa una discussione su come viene fissata tale soglia, nel capitolo 16.

Dobbiamo considerare le seguenti operazioni: (1) calcolare l'SVD [Singular Value Decomposition] della matrice, (2) definire una "tolleranza" (la soglia per identificare un numero come significativamente diverso da zero), (3) contare il numero di valori singolari al di sopra di questa tolleranza.

Geometria: difficoltà nel calcolare il rango geometricamente. Data una matrice 3×3 che rappresenta alcuni dati raccolti da un satellite. Le colonne sono in R^3. Supponiamo che i tre vettori giacciono tutti su un piano 2D --> il rango della matrice dei dati deve essere 2.

Ma i sensori del satellite non sono perfetti e c'è un piccolissimo rumore che corrompe il segnale. Quindi, in effetti, se guardassi il sottospazio attraversato dalle colonne della matrice a "altezza degli occhi", dovremmo vedere i vettori perfettamente disposti su un piano. Invece i vettori che puntano leggermente sopra o sotto quel piano. Il computer direbbe rango 3, ma sappiamo che in realtà esso è dovuto al rumore del sensore. Allora desideriamo ignorare una piccola quantità di rumore.



In [14]:
```
# zeros matrix
Z = np.zeros((5,5))

# tiny noise matrix
N = np.random.randn(5,5)

# add them
ZN = Z + N*np.finfo(float).eps*1e-307

# print the results
print('Rango di Z = ',np.linalg.matrix_rank(Z))
print('Rango di AZ = ',np.linalg.matrix_rank(ZN))
print('Norma di AZ = ',np.linalg.norm(ZN,'fro'))
```

```
Rango di Z =  0
Rango di AZ =  5
Norma di AZ =  0.0
```

Section 7.11, Rank in Span

Un vettore si trova nell'intervallo di un altro insieme di vettori **S** se **v** può essere scritto come una combinazione ponderata dei vettori di **S**.
Possiamo reinterpretare questo problema nel contesto delle matrici e del rango.

1. Inseriamo i vettori dell'insieme S in una matrice **S**.

2. Calcoliamp il rango di **S**. Chiamiamo quel rango r1.

3. Aumentiamo **S** di **v**, creando così **Sv** = **S** ⊔ **v**

4. Calcoliamo il rango di **Sv**. Chiamiamo rango r2.

   Se r2 > r1, --> **v** non è span si **S**.

   Se r2 = r1, --> **v** è span di **S**.

   Se r2 < r1, --> testare il codice, c'è un errore

In [ ]: