LAbook_chapter03

July 10, 2024

1 BOOK: Linear Algebra: Theory, Intuition, Code

AUTHOR: Mike X Cohen

WEBSITE: sincxpress.com

1.1 CHAPTER: Vector multiplications (chapter 3)

1.1.1 Section 3.1, Vector dot product

1.1.2 Import libraries for the entire chapter

[1]:
```
import numpy as np
import sympy as smp
from sympy.vector import *
from IPython.display import Image
```

1.1.3 Prodotto scalare "dot product" – il risultato è uno scalare

$$\alpha = \mathbf{a} \cdot \mathbf{b} = \langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^{\mathrm{T}}\mathbf{b} = \sum_{i=1}^{n} a_i b_i$$

$$[1\ 2\ 3\ 4] \cdot [5\ 6\ 7\ 8] = 1x5 + 2x6 + 3x7 + 4x8 = 5 + 12 + 21 + 32 = \mathbf{70}$$

$$\mathbf{a}^{\mathrm{T}}\mathbf{b} = \|\mathbf{a}\|\|\mathbf{b}\| \cos\theta_{ab}$$

1.1.4 Section 3.1, Vector dot product: Algebra

[2]:
```
Image("img/picture.png")

# two vectors
v1, v2, dp = smp.symbols('v1 v2 dp')
v1 = np.array([2,5,4,7])
v2 = np.array([4,1,0,2])

# dot product between them
dp = np.dot(v1,v2)
print(v1," = v1 -- ",v2," = v2 -- ",dp," = dp, dot product")
```

```
[2 5 4 7]  = v1 --  [4 1 0 2]  = v2 --  27  = dp, dot product
```

1.1.5 Section 3.2, Dot product properties

associative property – commutative property – distributive property – Cauchy-Schwarz ineuality

1.1.6 Section 3.3, Vector dot product: Geometry

1.1.7 Section 3.4, Algebraic and geometric equivalence

Proof of the Cauchy-Schwarz ineuality

1.1.8 Section 3.5, code block 3.3 - Linear wighted combination

1.1.9 Combinazione lineare pesata

$$\mathbf{w} = \lambda_1\mathbf{v}_1 + \lambda_2\mathbf{v}_2 + \cdots + \lambda_n\mathbf{v}_n$$

[7]:
```
# scalars
l1, l2, l3, v3 = smp.symbols('l1 l2 l3 v3')
l1 = 1
l2 = 2
l3 = -3

# vectors
v1 = np.array([4,5,1])
v2 = np.array([-4,0,-4])
v3 = np.array([1,3,2])

# linear weighted combinations
w = l1*v1 + l2*v2 + l3*v3
print(w," = w = linear weighted combination")
```

```
[ -7  -4 -13]  = w = linear weighted combination
```

1.1.10 Section 3.6, code block 3.5 - Outer product

Element perspective – row perspective

Prodotto esterno "outer product" $\begin{bmatrix} a \\ b \\ c \end{bmatrix} [d\ e\ f] = \begin{bmatrix} ad & ac & af \\ bd & bc & bf \\ cd & cc & cf \end{bmatrix}$

[8]:
```
# two vectors
op = smp.symbols('op ')
v1 = np.array([2,5,4,7])
v2 = np.array([4,1,0,2])

# outer product
op = np.outer(v1,v2)
print("",v1," = v1","\n\n",v2," = v2","\n\n",op," = op = prodotto esterno")
```

```
[2 5 4 7]  = v1

[4 1 0 2]  = v2

[[ 8  2  0  4]
[20  5  0 10]
[16  4  0  8]
[28  7  0 14]]  = op = prodotto esterno
```

1.1.11 Section 3.7, code block 3.7 - Element-wise (Hadamard) vector product

1.1.12 Prodotto vettoriale Hadamard – "Hadamard vector product"

$$\mathbf{c} = \mathbf{a} \odot \mathbf{b} = [a_1 b_1 \quad a_2 b_2 \ldots \ldots \ a_n b_n]$$

[9]:
```
# two vectors
v1 = np.array([2,5,4,7])
v2 = np.array([4,1,0,2])

# Hadamard
v3 = v1 * v2
print("" ,v1," = v1\n\n",v2," = v2\n\n",v3," = v3 = Hadamard vectorial product")
```

```
[2 5 4 7]  = v1

[4 1 0 2]  = v2

[ 8  5  0 14]  = v3 = Hadamard vectorial product
```

1.1.13 Secton 3.8, Cross product

1.1.14 Section 3.9, code block 3.9 - Unit vector

1.1.15 Vettore unitario – "Unit vector"

$$\hat{v} = \frac{1}{\|v\|} v = \frac{1}{\sqrt{\sum_{i=1}^{n} v_i^2}} v$$

[10]:
```
v, vMag,v_unit = smp.symbols('v vMag v_unit')
# a friendly vector
v = np.array([2,5,4,7])

# its norm
vMag = np.linalg.norm(v)

# the unit vector
v_unit = v / vMag
print("",v," = v\n\n",vMag," = vMag\n\n",v_unit," = v_unit = Unit vector")
```

```
[2 5 4 7]  = v

9.695359714832659  = vMag

[0.20628425 0.51571062 0.4125685  0.72199487]  = v_unit = Unit vector
```

1.1.16 Section 3.10, Exercises

1.1.17 Sestion 3.11, Answers

1.1.18 Section 3.12, Code challenges

1.1.19 Section 3.13, code block 3.11 – Code solutions

1.1.20 Media pesata di N numeri – Weigthed average of N numbers

[11]:
```
# some vectors
v1 = np.array([1,2,3,4,5])
v2 = np.array([2,3,4,5,6])
v3 = np.array([3,4,5,6,7])

# the weightings
w = [-1,3,-2]

# their weighted combo
result = v1*w[0] + v2*w[1] + v3*w[2]
```

1.1.21 Section 3.13, code block 3.13 – Average of N numbers

1.1.22 Media di N numeri – Average of N numbers

[12]:
```
o, s, ave = smp.symbols('v s ave')
# a vector (technically, type list)
v = [ 7, 4, -5, 8, 3 ]

# the ones vector
o = np.ones(len(v))
# sum of the vector elements
s = np.dot(v,o)
# dimensione del vettore
elle = len(v)
# average via dot product
ave = np.dot(v,o) / len(v)
print("",v," = v\n\n",elle," = elle = dimensione de vettore\n\n",s," = s = somma␣
 ↪degli elementi\n\n",ave," = ave = media degli elementi")
```

```
[7, 4, -5, 8, 3]  = v

5  = elle = dimensione de vettore
```

```
17.0  = s = somma degli elementi

3.4  = ave = media degli elementi
```

1.1.23 Section 3.13, code block 3.15 – Weighted average with randomized weights

1.1.24 Media pesata statistica di N numeri – Statistical weighted average of N numbers

[13]:
```
wAve = smp.symbols('wAve')
# list
v = [ 7, 4, -5, 8, 3 ]

# random weightings
w = np.random.rand(len(v))

# weighted dot product
wAve = np.dot(v, w/sum(w))
print("",v," = v vettore\n\n",w," = w = peso random\n\n",wAve," = wAve = valore␣
↪medio pesato statistico")
```

```
[7, 4, -5, 8, 3]  = v vettore

[0.18293869 0.62096743 0.33172969 0.9165119  0.85857746]  = w = peso random

4.1273630919815965  = wAve = valore medio pesato statistico
```

[ ]:
```

```