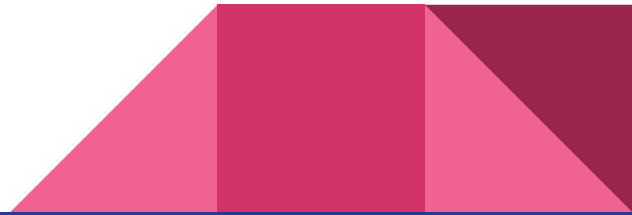
06. Operazioni in NumPy

Corso di Python per il Calcolo Scientifico

Angelo Cardellicchio
angelo.cardellicchio@stiima.cnr.it

Outline

- Algebra
- Polinomi
- Statistica



Algebra (1)

- Alcune delle funzioni per le operazioni algebriche sono contenute nel package **`linalg`**
- Per calcolare la trasposta di una matrice:

```
x = np.array([[1, 2, 3], [4, 5, 6]])
np.transpose(x)
```

- Per calcolare l'inversa di una matrice:

```
mat = np.array([[5, 0, 0], [0, 2, 0], [0, 0, 4]])
linalg.inv(mat)
```

Algebra (2)

- Per calcolare il **prodotto scalare**, definito come:

$$p = \sum_i v_{1i} \cdot v_{2i}$$

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
np.inner(a, b)
```

- Per calcolare il **prodotto esterno**, definito come:

$$P = \begin{pmatrix} a_1 \cdot b_1 & \cdots & a_1 \cdot b_n \\ \vdots & \ddots & \vdots \\ a_n \cdot b_1 & \cdots & a_n \cdot b_n \end{pmatrix}$$

```
np.outer(a, b)
array([[3, 4], [6, 8]])
```

Algebra (3)

- La funzione **`matmul`** ci permette di effettuare il prodotto matriciale:

```
a = np.array([[1, 2], [3, 4]])
b = np.array([[5, 6], [7, 8]])
np.matmul(a, b) array([[19, 22], [43, 50]])
```

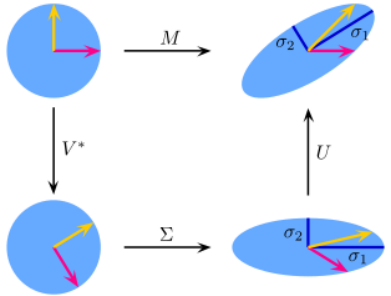
- Per calcolare la potenza di una matrice:

```
matrix_power(a, 5)
array([[1069, 1558], [2337, 3406]])
```

Algebra (4)

- È anche possibile effettuare la **decomposizione ai valori singolari**:

```
(u, s, v) = linalg.svd(a)
u: [[-0.40455358 -0.9145143 ] [-0.9145143
        0.40455358]]
s: [5.4649857 0.36596619]
v: [[-0.57604844 -0.81741556] [ 0.81741556 -
        0.57604844]]
```



$$M = U \cdot \Sigma \cdot V^*$$

•La decomposizione ai valori singolari è una tecnica di **riduzione della dimensionalità**

•Consiste nello scomporre la matrice iniziale in tre diverse sottomatrici

•Si basa sul concetto di **trasformazione lineare**

Algebra (5)

- È possibile calcolare **norma**, **determinante**, **rango** e **traccia**:

```
linalg.norm(mat)          # Il risultato sarà 16.97...
linalg.det(mat)           # Il risultato sarà 6
linalg.matrix_rank(mat) # Il risultato sarà 3
np.trace(mat)             # Il risultato sarà 20
```

- Infine, possiamo risolvere un **sistema di equazioni lineari**:

```
b = np.array([3, 2, 3])
linalg.solve(mat, b)
```

Polinomi (1)

- Per rappresentare un polinomio, usiamo un oggetto di classe **`poly1d()`**.
- Per sommare due polinomi, usiamo la funzione **`polyadd(p1, p2)`**:

```
c1 = (0, 2, 1)
c2 = (1, 3, 2)
poly.polyadd(c1, c2)
```

- Per sottrarre due polinomi, usiamo la funzione **`polysub(p1, p2)`**:

```
poly.polysub(c2, c1)
```

- Analogamente abbiamo **`polymul(p1, p2)`** (moltiplicazione), **`polydiv(p1, p2)`** (divisione) e **`polypow(p, pow)`** (elevazione a potenza).

Polinomi (2)

- Per caratterizzare il valore assunto da un polinomio usiamo la funzione **`polyval(vals, p)`**.
- Il calcolo della derivata di un polinomio avviene mediante la funzione **`polyder(p, o)`**, dove **`o`** è l'ordine della derivata.
- L'operazione duale alla derivazione, ovvero l'integrazione, viene effettuata mediante la funzione **`polyint(p, o)`**, dove o è l'ordine di integrazione.

Statistica (1)

- Il **q – percentile** di un vettore V di lunghezza N è definito come il valore pari a $\frac{q}{100}$ calcolato a partire da una copia ordinata di V.
- Ad esempio:

$$V = [1, 5, 6, 7]$$
$$50PC(V) = 5.5$$

- Esistono diversi modi di calcolare il q – percentile*.
- NumPy ci mette a disposizione la funzione **`percentile()`**:

```
np.percentile(v, 50)
```

- Il quantile è sostanzialmente analogo al percentile con valori normalizzati. Per calcolarlo ci offre la funzione **`quantile()`**:

```
np.quantile(v, .5)
```

***Hyndman, R. J., & Fan, Y. (1996). Sample quantiles in statistical packages. *The American Statistician*, *50*(4), 361-365.**

Statistica (2)

- La media degli elementi di un array può essere calcolata in due modi.
- Il primo prevede il calcolo puramente aritmetico mediante la funzione **`mean()`**:

```
a = np.array([5, 12, 22, 3])
np.mean(a)
10.5
```

- Il secondo usa una media pesata mediante la funzione **`average()`**:

```
np.average(a, weights=[3, 1, 1, 3])
np.mean(a)
7.25
```

- Esistono anche delle funzioni per calcolare deviazione standard e varianza:

```
np.std(a)
np.var(a)
```

Domande?

42