Numpy

Matej Oravec

BAB37ZPR

10. týždeň

Dnešná prednáška

Obsah

Dnešná prednáška

Úvod

Čo je Numpy

- Balíček na praktické a efektívne riešenie matematických problémov
- Funkcie „základnej vrstvy“ sú predkompilovaným kódom v jazyku C
  - Možnosť dosiahnuť omnoho vyššiu rýchlosť než v čistom Pythone
  - Vyžaduje to vektorizáciu výpočtov

Inštalácia

- `pip install numpy` ▸ PyPI
- Numpy je súčasťou inštalácie Anaconda
  - Ak máte nainštalovaný Anaconda Python, netreba riešiť nič

Zaužívaný import

- `import numpy as np`

`numpy.ndarray`

Základný dátový typ v Numpy

Najzákladnejšia možnosť vytvorenia

- Funkcia `numpy.array()`

Docs

Rozdiely oproti zoznamu

- Je *homogénne*
  - Dátový typ je pre každý prvok poľa rovnaký
- Pozná svoju *dimenzionalitu*

```
1 a1 = np.array([[1, 2], [3, 4]]) # 2D pole typu int64
2 a2 = np.array([[1], [2, 3]]) # 1D pole typu object
```

- Zjednodušené indexovanie
  - `a1[0, 1]`
  - `a1[:, -1]`

numpy.ndarray

Užitočné možnosti vytvorenia poľa

- `numpy.ones()`, `numpy.ones_like()` — Docs
  - Pole plné jednotiek
  - Špecifikované buď rozmermi (`ones`) alebo iným poľom (`ones_like`)
  - Podobne `zeros`/`zeros_like`, `empty`/`empty_like`, `full`/`full_like`
  - Default dátový typ je float64
- `numpy.arange()` — Docs
  - Rovnomerne rozložené hodnoty odtiaľ potiaľ s takým a takým krokom
  - Podobné ako štandardná funkcia `range()`
    - ★ range vracia *iterátor*, zatiaľčo `numpy.arange` vracia `numpy.ndarray`
- `numpy.linspace()` — Docs
  - Rovnomerne rozložené hodnoty na zadanom intervale
  - Špecifikované ich počtom, nie krokom
  - Podobné funkcie pre logaritmické delenie intervalu: `geomspace`, `logspace`

Vektorizácia

Univerzálne funkcie

O čo ide

- Funkcie, ktoré operujú rozdielnym spôsobom v závislosti od dátového typu objektu, na ktorý sú zavolané
- Podpora broadcastingu
  - Volanie funkcie na každý z elementov poľa

▸ Viac

```
>>> a = np.array([10, 20, 30])
>>> b = np.array([1, 2, 3]).reshape((3, 1))

>>> a + b
array([[11, 21, 31],
       [12, 22, 32],
       [13, 23, 33]])
```

Vektorizácia

Univerzálne funkcie

O čo ide

- Funkcie, ktoré operujú rozdielnym spôsobom v závislosti od dátového typu objektu, na ktorý sú zavolané
- Podpora broadcastingu
  - Volanie funkcie na každý z elementov poľa

▸ Viac

```
>>> a = np.array([1, 2, 3])

>>> 5 * a
array([ 5, 10, 15])

>>> a + 5
array([6, 7, 8])
```

Vektorizácia

Nahradenie cyklov

Príklad: Výpočet hodnôt funkcie sin()

- Úlohou je získať hodnoty funkcie sin() na rovnomernom delení intervalu $\langle 0, 2\pi \rangle$, obsahujúcom 100 bodov (okraje zahŕňajúc)

a) Bez použitia Numpy

```
1  import math
2
3  n_points = 100
4  step = (2 * math.pi - 0) / (n_points - 1)
5  x = []
6  y = []
7
8  for i in range(n_points):
9      x.append(i * step)
10 for i in x:
11     y.append(math.sin(x))
```

Príklad: Výpočet hodnôt funkcie sin()

- Úlohou je získať hodnoty funkcie sin() na rovnomernom delení intervalu $\langle 0, 2\pi \rangle$, obsahujúcom 100 bodov (okraje zahŕňajúc)

b) Bez použitia Numpy – trochu praktickejšie

```
1 import math
2
3 n_points = 100
4 step = (2 * math.pi - 0) / (n_points - 1)
5
6 x = [i * step for i in range(n_points)]
7 y = [math.sin(point) for point in x]
```

Príklad: Výpočet hodnôt funkcie sin()

- Úlohou je získať hodnoty funkcie sin() na rovnomernom delení intervalu $\langle 0, 2\pi \rangle$, obsahujúcom 100 bodov (okraje zahŕňajúc)

c) S použitím Numpy (bez cyklu!)

```
1 import numpy as np
2
3 x = np.linspace(0, 2 * np.pi, endpoint=True)
4 y = np.sin(x)
```

Vektorizácia

Čo si zapamätať

- Numpy má rýchlostnú výhodu, ale iba pri vektorizovaných výpočtoch
  - Pri použití cyklov na `ndarray` je program väčšinou *pomalší* než pri použití čistého Pythonu
- V mnohých prípadoch je vektorizácia prirodzená
  - Predchádzajúci príklad s výpočtom hodnôt funkcie sin()
- Vo veľa prípadoch intuícii pomôže považovať všetky polia za *matice*

Dimenzionalita polí

Prispôsobenie

Zistenie rozmerov

- `np.shape()`
  - Vracia *n*-ticu

- Zvýšenie dimenzionality
  a) Pomocou indexovania

```
>>> a = np.array([1, 2, 3])

>>> a[:, np.newaxis] # Rovnaké ako a[:, None]
array([[1],
       [2],
       [3]])

>>> a[np.newaxis, :] # Rovnaké ako a[None, :]
array([[1, 2, 3]])
```

Dimenzionalita polí

Prispôsobenie

Zistenie rozmerov

- `np.shape()`
  - Vracia *n*-ticu

- Zvýšenie dimenzionality
  b) Pomocou funkcií na to priamo určených

```
>>> a = np.array([1, 2, 3])

>>> np.atleast_2d(a).T
array([[1],
       [2],
       [3]])

>>> np.atleast_2d(a)
array([[1, 2, 3]])
```

Dimenzionalita polí

Prispôsobenie

Zistenie rozmerov

- `np.shape()`
  - Vracia *n*-ticu

- Zvýšenie dimenzionality
  c) Pomocou metódy `reshape`

```
>>> a = np.array([1, 2, 3])

>>> a.reshape((3, 1))
array([[1],
       [2],
       [3]])

>>> a.reshape((1, 3))
array([[1, 2, 3]])
```

Dimenzionalita polí

Prispôsobenie

Zistenie rozmerov

- `np.shape()`
  - Vracia *n*-ticu

- Zníženie dimenzionality
  a) Pomocou funkcie `np.ravel()`

```
>>> a = np.arange(15).reshape((3, 5))

>>> a.ravel()
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])

>>> np.ravel(a)
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

Dimenzionalita polí

Prispôsobenie

Zistenie rozmerov

- `np.shape()`
  - Vracia *n*-ticu

- Zníženie dimenzionality
  b) Pomocou metódy `reshape` a výberu

```
>>> a = np.arange(15).reshape((3, 5))

>>> a.reshape((1, 15))[0]
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

Dimenzionalita polí

Transpozícia, prevrátenie

Transpozícia

- Pomocou funkcie `np.transpose()`
- Každé pole má definovanú hodnotu atribútu `T`

```
>>> a = np.arange(1, 10).reshape((3, 3))

>>> a
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])

>>> a.T
array([[1, 4, 7],
       [2, 5, 8],
       [3, 6, 9]])
```

Dimenzionalita polí

Transpozícia, prevrátenie

Otočenie

- Pomocou funkcie `np.flip()`
- Je možné špecifikovať os, podľa ktorej má byť pole otočené

```
>>> a = np.arange(1, 10).reshape((3, 3))

>>> a
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])

>>> np.flip(a, axis=0)
array([[7, 8, 9],
       [4, 5, 6],
       [1, 2, 3]])
```

Ďakujem za pozornosť.