Python cho Khoa học dữ liệu

Bài 17: Tổng quan về Thư viện Numpy

Hà Minh Tuấn
hmtuan@hcmus.edu.vn

Khoa Toán - Tin học
Trường Đại học Khoa học Tự nhiên, ĐHQG-HCM

Ngày 4 tháng 9 năm 2025

Nội dung bài học

NumPy là gì?

- **NumPy** (Numerical Python) là thư viện cốt lõi của Python cho tính toán khoa học.
- Cung cấp cấu trúc dữ liệu `ndarray` – mảng đa chiều mạnh mẽ và hiệu quả.
- Được tối ưu hóa cao, nhanh hơn danh sách Python thuần túy khi xử lý dữ liệu số lớn.
- Là nền tảng cho nhiều thư viện khoa học dữ liệu khác như: `Pandas`, `SciPy`, `Scikit-Learn`, `TensorFlow`.

Tạo mảng NumPy

- `numpy.array()` để tạo mảng từ danh sách.
- Các hàm tiện ích: `np.zeros()`, `np.ones()`, `np.arange()`, `np.linspace()`.

Ví dụ: Tạo mảng NumPy

```python
import numpy as np

a = np.array([1, 2, 3, 4, 5])
b = np.zeros((2, 3))
c = np.ones((3, 3))
d = np.arange(0, 10, 2)
e = np.linspace(0, 1, 5)

print(a)
print(b)
print(c)
print(d)
print(e)
```

```
[1 2 3 4 5]
[[0. 0. 0.]
 [0. 0. 0.]]
[[1. 1. 1.]
```

Indexing và Slicing

- Truy cập phần tử qua chỉ số: `a[0]`, `a[1]`.
- Lấy lát cắt: `a[start:end:step]`.
- Hỗ trợ indexing đa chiều.

Ví dụ: Indexing và Slicing

```
x = np.array([[1,2,3],[4,5,6],[7,8,9]])

print(x[0,0])     # phần tử (0,0)
print(x[1,:])     # hàng 1
print(x[:,2])     # cột 2
print(x[0:2, 1:3]) # sub-matrix
```

```
1
[4 5 6]
[3 6 9]
[[2 3]
 [5 6]]
```

Các phép toán cơ bản

- Phép toán phần tử: cộng, trừ, nhân, chia.
- Các hàm toán học: `np.sqrt()`, `np.exp()`, `np.sin()`.
- Tính tổng, trung bình, min, max: `np.sum()`, `np.mean()`, `np.min()`, `np.max()`.

Ví dụ: Các phép toán

```
arr = np.array([1,2,3,4,5])

print(arr + 10)
print(arr * 2)
print(np.sqrt(arr))
print(np.mean(arr))
```

```
[11 12 13 14 15]
[ 2  4  6  8 10]
[1.         1.41421356 1.73205081 2.         2.23606798]
3.0
```

Khái niệm Broadcasting

- Cho phép thực hiện phép toán giữa mảng có kích thước khác nhau.
- NumPy sẽ tự động "mở rộng"mảng nhỏ để phù hợp với mảng lớn.
- Rất hữu ích trong xử lý dữ liệu lớn mà không cần vòng lặp.

Ví dụ: Broadcasting

```
A = np.array([[1,2,3],[4,5,6],[7,8,9]])
b = np.array([1,0,1])

print(A + b)
```

```
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]]
```

Ứng dụng NumPy trong Khoa học Dữ liệu

- Tiền xử lý dữ liệu (chuẩn hóa, biến đổi).
- Tính toán thống kê, ma trận, đại số tuyến tính.
- Xây dựng nền tảng cho Machine Learning (ML).
- Hỗ trợ xử lý dữ liệu ảnh, tín hiệu.

Đại số tuyến tính trong NumPy

- NumPy hỗ trợ tính toán ma trận và đại số tuyến tính qua module `numpy.linalg`.
- Một số hàm quan trọng:
  - `np.dot(A, B)`: tích ma trận.
  - `np.linalg.inv(A)`: nghịch đảo ma trận.
  - `np.linalg.eig(A)`: trị riêng và vector riêng.

Ví dụ: Đại số tuyến tính

```
A = np.array([[1, 2],
              [3, 4]])

print("det(A) =", np.linalg.det(A))
print("inv(A) =", np.linalg.inv(A))
vals, vecs = np.linalg.eig(A)
print("Eigenvalues:", vals)
```

```
det(A) = -2.0000000000000004
inv(A) = [[-2.   1. ]
          [ 1.5 -0.5]]
Eigenvalues: [-0.37228132  5.37228132]
```

Ngẫu nhiên trong NumPy

- NumPy cung cấp module `numpy.random` để sinh số ngẫu nhiên.
- Các hàm thường dùng:
  - `np.random.rand()` – phân phối đều [0,1).
  - `np.random.randn()` – phân phối chuẩn.
  - `np.random.randint()` – sinh số nguyên ngẫu nhiên.

Ví dụ: Sinh số ngẫu nhiên

```
print(np.random.rand(3))       # 3 số thực đều
print(np.random.randn(3,3))    # ma trận chuẩn
print(np.random.randint(1,10,5)) # 5 số nguyên
```

```
[0.52 0.31 0.87]
[[-0.11  0.44 -1.23]
 [ 0.65 -0.54  0.90]
 [ 1.23 -0.98  0.17]]
[3 9 2 7 5]
```

Hiệu năng NumPy so với Python thuần

- NumPy sử dụng mảng đồng nhất, được cài đặt bằng C, nên tính toán nhanh hơn nhiều so với danh sách Python.
- Tránh sử dụng vòng lặp thuần trong Python khi làm việc với dữ liệu lớn.
- Sử dụng phép toán vector hóa trong NumPy để tối ưu hiệu năng.

Ví dụ: So sánh hiệu năng

```
import time

N = 10**6
a = list(range(N))
b = list(range(N))

start = time.time()
c = [a[i] + b[i] for i in range(N)]
print("Python list:", time.time()-start)

A = np.arange(N)
B = np.arange(N)

start = time.time()
C = A + B
print("NumPy array:", time.time()-start)
```

```
Python list: 0.18 s
```