D07 NumPy 的矩陣函式與線性代數應用

重要知識點

行介紹。

主題	函式
矩陣乘積	點積、內積、外積、矩陣乘法
矩陣操作	跡、行列式、反矩陣、轉置、特徵值與特徵向量、秩、線性系統求解
特殊矩陣	單位矩陣(identity)、單位矩陣(eye)、三角矩陣、單對角陣列、上三角矩陣、下三角矩陣
矩陣分解 (Matrix Decomposition)	Cholesky、QR、SVD

NumPy 線性代數函式

- numpy.linalg 是 NumPy 線性代數模組，提供線性代數函式相關的函式。
- NumPy 線性代數函式提供高效能的線性代數演算法，並且可透 OpenBLAS、MKL、ATLAS 等程式庫進行多線程的運算。
- 除了 NumPy 之外，SciPy 提供更多線性代數函式，可以搭配應用。

NumPy 矩陣乘積 – 點積 (dot product)

- 進行點積運算須注意形狀必須注意形狀 (shape)。若是兩個向量的點積，兩個向量的元素數目也須相同，或其中一個數目為 1 (廣播)。
- 若是兩個多維陣列 (矩陣) 的點積，則中間兩個大小要相同才能進行點積，例如：(2,3)·(3,4)→ 成為 (2,4)

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \end{bmatrix}$$

$$A \cdot B = \begin{bmatrix} a_{11} * b_{11} + a_{12} * b_{21} + a_{13} * b_{31} & a_{11} * b_{12} + a_{12} * b_{22} + a_{13} * b_{32} & a_{11} * b_{13} + a_{12} * b_{23} + a_{13} * b_{33} & a_{11} * b_{14} + a_{12} * b_{24} + a_{13} * b_{34} \\ a_{21} * b_{11} + a_{22} * b_{21} + a_{23} * b_{31} & a_{21} * b_{12} + a_{22} * b_{22} + a_{23} * b_{32} & a_{21} * b_{13} + a_{22} * b_{23} + a_{23} * b_{33} & a_{21} * b_{14} + a_{22} * b_{24} + a_{23} * b_{34} \end{bmatrix}$$

- NumPy點積函式語法：numpy.dot(a, b)

NumPy 矩陣乘積 – 內積 (inner product)

一般來說內積是 2 個一維陣列(向量)做內積。內積運算公式如下圖：

$$\vec{a} = [a_0, a_1, \ldots, a_{j-1}]$$

$$\vec{b} = [b_0, b_1, \ldots, b_{j-1}]$$

$$\vec{a} \cdot \vec{b} = \sum a_0 b_0 + a_1 b_1 + \ldots + a_{j-1} b_{j-1}$$

- 如果是多維陣列做乘積，那麼最後維度必須要符合，而內積結果就是其元素相乘之和。
- NumPy 內積函式語法：numpy.inner(a, b)

NumPy 矩陣乘積 – 外積 (outer product)

- 在線性代數中外積也稱為張量積。
- 外積是 2 個一維陣列(向量)做計算，若非一維陣列的話會先進行展平(flatten)再進行外積。公式及範例如右圖：

$$b = [b_0, b_1, \ldots, b_{j-1}]$$

$$\vec{a} \otimes \vec{b} = \begin{bmatrix} a_0 b_0 & a_0 b_1 & \ldots & a_0 b_{j-1} \\ a_1 b_0 & a_1 b_1 & \ldots & a_1 b_{j-1} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ a_{i-1} b_0 & a_{i-1} b_1 & \ldots & a_{i-1} b_{j-1} \end{bmatrix}$$

```
np.outer(a, b)
```

NumPy 矩陣乘法 – matmul 與 @

matmul 與 dot 都是矩陣乘法 (Matrix Multiplication)，兩者非常類似，其相同及不同點如下：

- 如果 2 個都是二維陣列的話，matmul 與 dot 相同。
- 在 matmul 中，多維的矩陣，將前 n-2 維視為後 2 维的元素後，進行乘法運算。
- matmul 不允許矩陣與純量相乘。

在 Python 3.5 版本之後提供 @ 做為矩陣相乘運算子，在多個矩陣相乘時提供更簡潔的語法。

matmul 與 @ 的計算結果完全相同。

```
np.matmul(A, B)
```

```
array([[20, 23, 26, 29],
       [56, 68, 80, 92]])
```

```
A @ B
```

```
array([[20, 23, 26, 29],
       [56, 68, 80, 92]])
```

NumPy 矩陣操作 – 跡(trace)

跡的運算說明如下圖：

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

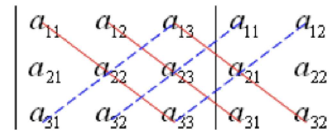
$$trace(A) = a_{11} + a_{22} + a_{33}$$

跡的運算可以透過 trace() 函式求得：np.trace(A)

NumPy 矩陣操作 – 行列式 (determinant)

行列式的運算說明如下圖：

$$= a_{11}\,a_{22}\,a_{33} + a_{12}\,a_{23}\,a_{31} + a_{13}\,a_{21}\,a_{32} - a_{13}\,a_{22}\,a_{31} - a_{23}\,a_{32}\,a_{11} - a_{33}\,a_{12}\,a_{21}$$



圖解行列式運算 (Source: Wikipedia)

行列式的運算可以透過 numpy.linalg.det() 函式求得：np.linalg.det(A)

NumPy 矩陣操作 – 反矩陣(inverse)

反矩陣的運算說明如下圖：

以三階矩陣為例：

$$A = \left[\begin{array}{ccc|ccc} a_{11} & a_{12} & a_{13} & 1 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 1 & 0 \\ a_{31} & a_{32} & a_{33} & 0 & 0 & 1 \end{array}\right]$$

透過高斯約當法 (Gaussian Elimination)，可以得到右邊的矩陣即為 A 的反矩陣

$$A^{-1} = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & b_{11} & b_{12} & b_{13} \\ 0 & 1 & 0 & b_{21} & b_{22} & b_{23} \\ 0 & 0 & 1 & b_{31} & b_{32} & b_{33} \end{array}\right]$$

反矩陣的運算可以透過 numpy.linalg.inv() 函式求得：np.linalg.inv(A)

NumPy 矩陣操作 – 轉置 (Transpose)

轉置的運算說明如下圖：

$$A = \begin{bmatrix} a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$A^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$

轉置矩陣可透過呼叫 numpy.transpose() 函式達成。

NumPy 矩陣操作 – 特徵值與特徵向量(eig)

eig() 函式可以用來計算方陣的特徵值(eigenvalue)與特徵向量(eigenvector)。
eig() 回傳值有2個：eigenvalues, eigenvectors。

另外 eigvals() 函式也可以計算特徵值。

```
# eigenvalues
w
```

```
array([-5.,  3.,  6.])
```

```
# eigenvectors
```

NumPy 矩陣操作 – 秩(rank)

呼叫 matrix_rank() 回傳矩陣的秩(rank)。

NumPy是使用SVD (奇異值分解；singular value decomposition) 來計算秩。

NumPy 矩陣操作 – 線性系統求解(solve)

求解線性系統可以使用 solve() 函式。

NumPy 特殊矩陣 – 單位矩陣(identity)

單位矩陣是對角線元素值為 1 的方陣。

NumPy 特殊矩陣 – 單位矩陣(eye)

eye() 也是用來產生對角線元素值為 1 的陣列，與 identity() 不同的地方在於 eye() 產生的可以不是方陣，也可以指定對角線開始的索引。
範例：

NumPy 特殊矩陣 – 單對角陣列(Diagonal)

diagonal() 與 diag() 語法基本上功能相同，差別在於 diagonal() 可以另行指定軸做為第一個軸或第二個軸。

也可以使用 array.diagonal() 語法。

NumPy 特殊矩陣 – 三角矩陣(tri)

開始的對角線以下元素值均為 1。

NumPy 特殊矩陣 – 上三角矩陣 (Upper Triangular)、下三角矩陣 (Lower Triangular)

triu()、numpy.tril() 可將對角線以上或以下的元素值設為 0，也可以指定對角線起始的索引。

NumPy 矩陣分解 (Matrix Decomposition)

NumPy 線性代數函式中提供3種矩陣分解的函式：

- Cholskey Decomposition
- QR Factorization
- Singular Value Decomposition (SVD)

NumPy 矩陣分解 - Cholskey Decomposition

Cholesky分解語法：numpy.linalg.cholesky()
範例：

QR分解語法：numpy.linalg.qr()

範例：

NumPy 矩陣分解 – SVD分解

SVD分解語法：numpy.linalg.svd()

範例：

知識點回顧

- 在今天的內容中介紹 NumPy 線性代數模組，示範矩陣的建立、操作、eigenvalues / eigenvectors、特殊矩陣、以及矩陣分解，需要用到線性代數相關運算的話，可以應用這些函式。
- 另外在範例中使用 LaTeX 撰寫矩陣範例，有興趣的話可以在 Jupyter Notebook 中查看語法。

延伸閱讀

NumPy Linear algebra (numpy.linalg) 官方文件

網站：[numpy]

下一步：閱讀範例與完成作業