Linear Algebra – Đại Số Tuyến Tính:

1. Cách Import?

from numpy import linalg as la

1. Trả Về Eigen Vector Và Eigen Value Của Ma Trận?

<Eigen Value>, <Eigen Vector> = la.eig(<Ma Trận Vuông>)

* <Ma Trận Vuông> và <Eigen Vector> là Tensor phức cùng Shape có 2 chiều cuối cùng có kích thước bằng nhau, 2 chiều này coi là 1 ma trận, chiều cuối cùng là chiều ngang, chiều kế cuối là chiều dọc
* Mỗi ma trận trong <Eigen Vector> có các cột là Eigen Vector độc lập tuyến tính và có độ dài = 1 của ma trận tương ứng trong <Ma Trận Vuông>
* <Eigen Value> là Tensor phức mà chiều cuối cùng của nó chứa Eigen Value của các Eigen Vector trong ma trận tương ứng trong <Eigen Vector>
* Các Eigen Vector sắp xếp không theo thứ tự nào
* Nếu ma trận không khả chéo hóa trong trường số phức, thì nó sẽ vẫn trả về Eigen Vector và Eigen Value nhưng bị sai

1. Trả Về Nghịch Đảo Của Ma Trận?

<Nghịch Đảo> = la.inv(<Ma Trận Vuông>)

1. Trả Về Lũy Thừa Của Ma Trận?

<Lũy Thừa> = la.matrix\_power(<Ma Trận Vuông>)

1. Hồi Quy Tuyến Tính?

<X>, <Loss>, <Hạng Của A>, <Singular Value Của A> = la.lstsq(

<A>, <B>, rcond = None

)

* <X> là giải pháp bình phương tối thiểu của phương trình <A><X> = <B> nếu phương trình này vô nghiệm, nếu có nghiệm, thì <X> là nghiệm đó luôn
* <A> là ma trận kích thước m x n, <B> là ma trận kích thước m x k, <X> là ma trận kích thước n x k
* <Loss> là 1 Numpy Array với Shape (k), phần tử thứ i = cột thứ i của B trừ cột thứ i của <A><X>, rồi bình phương, rồi cộng hợp lại
* <Hạng Của A> là 1 số nguyên
* <Singular Value Của A> là 1D Numpy Array có các phần tử là Singular Value của ma trận <A>

1. Tính Norm?

<Norm> = la.norm(

<Numpy Array>, ord = <Loại Norm>,

axis = <Trục>, keepdims = <Có Giữ Chiều Không>

)

* Mặc định

|  |  |
| --- | --- |
| <Loại Norm> | None |
| <Trục> | None |
| <Có Giữ Chiều Không> | False |

* Xét trường hợp <Trục> = None

|  |  |  |
| --- | --- | --- |
| <Loại Norm> | <Numpy Array> Shape | Cách tính |
| None | Bất kì | Tính bình phương Module mỗi phần tử trong <Numpy Array>, rồi cộng tất cả lại, rồi lấy căn bậc 2, nghĩa là dùng Euclidean Norm |
| 'f' hoặc 'fro' | 2D |
| 2 | 1D |
| Các trường hợp còn lại | Bất kì | <Trục> sẽ được gán lại thành Tuple có giá trị từ 0 đến số chiều của <Numpy Array> – 1, ví dụ (0, 1, 2), rồi chuyển tới giai đoạn bên dưới |

* Trường hợp <Trục> là 1 số, chỉ định chiều tính Norm

|  |  |
| --- | --- |
| <Loại Norm> | Cách tính |
| np.inf | Tính Module mỗi phần tử trong <Numpy Array>, rồi lấy Max theo <Trục>, tức là Maximum Norm |
| -np.inf | Như np.inf, thay chữ “Max” thành “Min”, tức là Minimum Norm |
| 0 | Đếm số phần tử khác 0 theo <Trục>, tức là Zero Norm |
| 1 | Tính Module mỗi phần tử trong <Numpy Array>, rồi tính tổng theo <Trục> |
| None hoặc 2 | Tính bình phương Module mỗi phần tử trong <Numpy Array>, rồi tính tổng theo <Trục>, rồi lấy căn bậc 2, tức là Euclidean Norm |
| <P> khác 0, 1, 2 | Lấy Module mỗi phần tử trong <Numpy Array>, rồi lũy thừa <P>, rồi tính tổng theo <Trục>, rồi lũy thừa 1 / <P>, tức là P Norm |

* Trường hợp <Trục> là 1 Tuple gồm 2 số, chỉ định trục cao và trục ngang của ma trận

|  |  |
| --- | --- |
| <Loại Norm> | Cách tính |
| 2 | Singular Value lớn nhất ứng với mỗi ma trận, tức là Spectral Norm |
| -2 | Singular Value bé nhất ứng với mỗi ma trận |
| 1 | Tính Module mỗi phần tử trong <Numpy Array>, rồi tính tổng mỗi cột của ma trận, lấy Max trong các tổng này, tức là Matrix Norm tạo ra từ P Norm với p = 1 |
| -1 | Như 1, thay chữ “Max” thành “Min” |
| np.inf | Như 1, thay chữ “cột” thành “hàng”, tức là Matrix Norm tạo ra từ P Norm với p = vô cùng |
| -np.inf | Như 1, thay chữ “Max” thành “Min”, “cột” thành “hàng” |
| None hoặc 'f' hoặc 'fro' | Lấy bình phương Module mỗi phần tử trong ma trận, cộng hết lại, rồi lấy căn bậc 2, tức là Frobenius Norm |
| 'nuc' | Tính các Singular Value của từng ma trận, rồi cộng hết lại, tức là Nuclear Norm |

Numpy:

1. Cách Import?

import numpy as np

1. Keep Dimension?

* Các hàm trong Numpy sẽ có thêm tham số keepdims, mặc định nó = False, nghĩa là, khi 1 hàm trong Numpy hoạt động trên 1 số chiều nào đó của Input, và những chiều này bị sụp về kích thước 1, thì kết quả cuối cùng sẽ loại bỏ những chiều này, ngược lại nếu keepdims = True, thì vẫn giữ những chiều này
* Numpy Array có thể có Shape = (), khi này nó chỉ là 1 con số

1. Nhân 2 Ma Trận?

<Kết Quả> = <Ma Trận 1> @ <Ma Trận 2>

1. Nhân Từng Phần Tử Trong Mảng?

<Kết Quả> = np.multiply(<Mảng 1>, <Mảng 2>)

* Ví dụ

foo = np.multiply([1, 2, 3], [4, 5, 6])

* “foo” có giá trị là [4, 10, 18]

1. Chồng Các Numpy Array Lên Nhau?

<Numpy Array Chồng> = np.stack(<Các Numpy Array>, <Chiều Quay Mặt>)

* Mặc định

|  |  |
| --- | --- |
| <Chiều Quay Mặt> | 0 |

* Hoạt động y chang cách chồng các Tensor lên nhau

1. Liệt Kê Tất Cả Các Giá Trị Khác Nhau Của 1 Numpy Array Theo Thứ Tự Tăng Dần?

<Các Giá Trị> = np.unique(<Numpy Array>)

* Ví dụ

foo = np.unique(bar)

* Ta có

|  |  |
| --- | --- |
| bar | [[1, 1, 2, 2, 5, 4,  9, 9, 4, 4, 3, 2]] |
| foo | [1, 2, 3, 4, 5, 9] |

1. Tất Cả Kiểu Dữ Liệu?

* Số nguyên

np.<Có Dấu Không>int<Số Bit>

* <Số Bit> = 8, 16, 32, 64
* <Có Dấu Không> = u thì không dấu, mặc định có dấu
* Số Floating Point

np.float<Số Bit>

* <Số Bit> = 16, 32, 64, 128
* Logic

bool

1. Concat 2 Ma Trận?

* Để trả về Concat của ma trận A với ma trận B theo chiều ngang, nghĩa là A kích thước m x n, Concat với B kích thước k x n, sẽ được ma trận kích thước

(m + k) x n

np.c\_[<A>, <B>]

1. Trả Về Tensor Chỉ Toàn 1?

np.ones(<Shape>, dtype = <Kiểu Dữ Liệu>)

* Mặc định

|  |  |
| --- | --- |
| <Kiểu Dữ Liệu> | np.float64 |

* <Shape> phải là Tuple nếu số chiều từ 2 trở lên, còn nếu số chiều = 1, thì chỉ cần là 1 số nguyên
* Ví dụ

foo = np.ones((4, 5))

1. Tính Variance?

<Variance> = np.var(

<A>, axis = <Chiều Tính>,

ddof = <Hiệu Chỉnh>, keepdims = <Có Giữ Chiều Không>

)

* Mặc định

|  |  |
| --- | --- |
| <Chiều Tính> | None |
| <Hiệu Chỉnh> | 0 |
| <Có Giữ Chiều Không> | False |

* Nếu <Chiều Tính> = None, thì sẽ tính Variance của nguyên Tensor <A>, nghĩa là lấy mỗi phần tử trong <A>, trừ đi Mean của <A>, bình phương lên, rồi cộng tất cả lại với nhau, rồi chia (số phần tử của <A> – <Hiệu Chỉnh>)
* Nếu chỉ định <Chiều Tính>, thì sẽ tính theo chiều đấy

1. Tính Ma Trận Hiệp Phương Sai Chéo?

<Covariance Matrix> = np.cov(

<X>, <Y>, rowvar = <Chiều Biến>,

bias = <Có Lệch Không>, ddof = <Hiệu Chỉnh>,

fweights = <Số Lần Xuất Hiện>, aweights = <Trọng Số Phần Tử>

)

* Mặc định

|  |  |  |  |
| --- | --- | --- | --- |
| <Chiều Biến> | True | <Có Lệch Không> | False |
| <Hiệu Chỉnh> | None | <Số Lần Xuất Hiện> | 1 |
| <Trọng Số Phần Tử> | 1 |  |  |

* Nếu <Chiều Biến> = True, thì mỗi cột của <X> và <Y> là 1 điểm, Concat <X> và <Y> theo chiều ngang được ma trận kích thước m x n, Plot m điểm này vào không gian n chiều, ngược lại, nếu = False, thì mỗi hàng của <X> và <Y> là 1 điểm, Concat <X> và <Y> theo chiều dọc được ma trận kích thước m x n, Plot n điểm này vào không gian m chiều
* Sau khi các điểm được Plot, thì mỗi điểm sẽ được phân thân với số phân thân =

<Số Lần Xuất Hiện> \* <Trọng Số Phần Tử>, số lần phân thân có thể không nguyên, <Số Lần Xuất Hiện> và <Trọng Số Phần Tử> đều phải có Shape là

(<Số Điểm>), sau đó tính trung bình tọa độ của toàn bộ các điểm, kể cả các điểm phân thân, ta được tọa độ M

* Tiến hành phân thân lại, lần này không tính <Trọng Số Phần Tử>, mà chỉ phân thân với số lần được ghi trong <Số Lần Xuất Hiện>, lấy tọa độ mỗi điểm, kể cả các điểm phân thân, trừ đi M, ta được tọa độ mới của các điểm, sắp xếp các tọa độ mới này thành các cột, ta được ma trận N, lấy hàng thứ i của N nhân vô hướng với hàng thứ j của N, rồi chia cho (<Số Điểm Kể Cả Phân Thân> – <Hiệu Chỉnh>), ta được phần tử hàng i cột j trong <Covariance Matrix>, nếu không chỉ định <Hiệu Chỉnh>, thì nếu <Có Lệch Không> = False, <Hiệu Chỉnh> sẽ = 1, còn nếu = True, thì <Hiệu Chỉnh> = 0

1. Tính Ma Trận Hệ Số Tương Quan?

<Correlation Matrix> = np.corrcoef(<X>, rowvar = <Chiều Biến>)

* Mặc định

|  |  |
| --- | --- |
| <Chiều Biến> | True |

* Cơ chế tính tương tự khi tính ma trận hiệp phương sai chéo, dựa theo công thức hệ số tương quan Pearson, với Covariance và Variance không được hiệu chỉnh

1. Trả Về String Chứa Mã Nguồn Của 1 Lệnh?

np.source(<Lệnh>)

* Ví dụ

np.source(np.cov)

1. Tạo Ma Trận Đường Chéo?

<Ma Trận Đường Chéo> = np.diag(<Đường Chéo>, <Index>)

* Mặc định

|  |  |
| --- | --- |
| <Index> | 0 |

* Các trường hợp

|  |  |
| --- | --- |
| <Đường Chéo> Shape | Cách tạo <Ma Trận Đường Chéo> |
| (<Số Phần Tử>) | Đặt lần lượt các phần tử trong <Đường Chéo> vào đường chéo Offset <Index> của ma trận không kích thước (<Số Phần Tử> + |<Index>|) x  (<Số Phần Tử> + |<Index>|) |
| (<M>, <N>) | Liệt kê lần lượt các phần tử trên đường chéo Offset <Index> của <Đường Chéo> vào 1 1D Numpy Array |

Numpy Array:

1. Numpy Array Chỉ Chạy Trên CPU?

* Đúng

1. Cách Nhanh Nhất Để Đổi Kiểu Dữ Liệu Của 1 Numpy Array?

<Numpy Array>.dtype = <Kiểu Dữ Liệu>

* Ví dụ

foo.dtype = float

* Để Copy ra 1 Numpy Array mới

<Numpy Array Mới> = <Numpy Array>.astype(<Kiểu Dữ Liệu>)

1. Chuyển Đổi PIL Image Sang Numpy Array?

<Numpy Array> = np.array(<PIL Image>)

1. Cách Hoán Vị Shape Của 1 Numpy Array?

<Numpy Array Có Shape Hoán Vị> = <Numpy Array Gốc>.transpose(<Hoán Vị>)

* Hoạt động đéo khác gì hoán vị Shape của 1 Tensor

1. Cách Gộp Tổng Giá Trị Vào 1 Trục?

<Numpy Array Ép> = <Numpy Array Gốc>.sum(

<Chiều Không Gian>, keepdims = True

)

* Hoạt động y chang như trong Torch

1. Trả Về Phiên Bản Numpy Array Bị Làm Phẳng?

<Numpy Array Phẳng> = <Numpy Array>.ravel(<Thứ Tự>)

* Mặc định

|  |  |
| --- | --- |
| <Thứ Tự> | 'C' |

* Bảng hoạt động

|  |  |
| --- | --- |
| <Thứ Tự> |  |
| 'C' | <Numpy Array Phẳng> dùng chung dữ liệu với <Numpy Array>, thứ tự đọc là từ chiều cuối tới chiều đầu, ví dụ 000, 001, 002, 010, 011, 012, … |
| 'F' | <Numpy Array Phẳng> không dùng chung dữ liệu với <Numpy Array>, thứ tự đọc là từ chiều đầu tới chiều cuối, ví dụ 000, 100, 200, 010, 110, 210, … |

Random – Ngẫu Nhiên:

1. Cách Import?

from numpy import random

1. Chỉ Định Seed?

random.seed(<Số Nguyên>)

1. Tạo Numpy Array Có Giá Trị Ngẫu Nhiên Thuộc Phân Phối Chuẩn Tắc?

random.randn(<Shape>)

* Hoạt động đéo khác gì torch.randn

1. Tạo Numpy Array Có Giá Trị Là Số Nguyên Ngẫu Nhiên?

random.randint(<Min>, <Max>, <Shape>)

* Hoạt động đéo khác gì torch.randint

Numpy

.stack(Arrays, Axis):

Stack các Array lên nhau

Ex:

numpy.stack((,)) =

.concatnate(Arrays, Axis):

Concat các Array theo Axis

Ex:

numpy.concatnate((), axis = 0)

=

.linspace(Low, High, Num):

Trả về Array nội suy tuyến tính từ Low đến High gồm Num phần tử (đã bao

gồm Low và High)

Ex:

numpy.linspace(2, 6, 5) = [2, 3, 4, 5, 6]

.meshgrid(X, Y):

Trả về 1 Array gồm 2 phần tử, một là 2D Array có các Row = X, hai là 2D

array có các Column = Y, kết hợp mỗi phần tử tương ứng trong X với Y tạo

thành Grid

Ex:

X = [1, 2, 3], Y = [4, 5, 6]

X, Y = numpy.meshgrid(X, Y)

X = , Y = , (X, Y) =

array

.ndim :

Trả về số Dimension của Array

Ex:

[[1]].ndim = 2

.repeat(Time, Axis):

Lặp lại Array Time lần theo Axis

Ex:

numpy.repeat(, 2, axis = 0) =

.argsort():

Trả về Array gồm các Index đã được sắp xếp sao cho giá trị Array tăng dần

Ex:

array = [3, 7, 2, 4, 8]

indices = array.argsort() = [2, 0, 3, 1, 4]

array[indices] = [2, 3, 4, 7, 8]

linalg

.eig(Matrix):

Trả về Array gồm 2 phần tử, một là Eigen Values List, hai là Normalized

Eigen Vectors Matrix, không theo thứ tự

Ex:

numpy.linalg.eig() =

.svd(Matrix):

Trả về Tuple gồm 3 phần tử, một là Left Singular Vectors Matrix, hai là

Singular Values Vector, ba là Transposed Right Singular Vectors Matrix

Ex:

numpy.linalg.svd () =

random

.exponential(Scale, Size):

Trả về Array là Samples từ Exponential Distribution với độ rộng Scale

Ex:

numpy.random.exponential(5, (3, )) =