NUMPY MATRIX AND LINEAR ALGEBRA

difference between numpy dot() and inner()

What's difference between numpy dot() and inner()?

Let's look into 2D array as an example:

```
>>> a=np.array([[1,2],[3,4]])
>>> b=np.array([[11,12],[13,14]])
>>> np.dot(a,b)
array([[37, 40],
       [85, 92]])
>>> np.inner(a,b)
array([[35, 41],
       [81, 95]])
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 11 & 12 \\ 13 & 14 \end{bmatrix}$$

With **dot()**:

$$\begin{bmatrix} 1*11+2*13 & 1*12+2*14 \\ 3*11+4*13 & 3*12+4*14 \end{bmatrix} = \begin{bmatrix} 37 & 40 \\ 85 & 92 \end{bmatrix}$$

With **inner()**:

$$\begin{bmatrix} 1*11+2*12 & 1*13+2*14 \\ 3*11+4*12 & 3*13+4*14 \end{bmatrix} = \begin{bmatrix} 35 & 41 \\ 81 & 95 \end{bmatrix}$$

NumPy Matrix

The chapters on NumPy have been using arrays (NumPy Array Basics A (python_numpy_array_tutorial_basic_A.php) and NumPy Array Basics B (python_numpy_array_tutorial_basic_B.php)). However, for certain areas such as linear algebra, we may instead want to use matrix.

```
>>> import numpy as np
>>> A = np.matrix([[1.,2], [3,4], [5,6]])
>>> A
matrix([[ 1.,  2.],
        [ 3.,  4.],
        [ 5.,  6.]])
```

We may also take the Matlab style by giving a string rather than a list:

```
>>> B = np.matrix("1.,2; 3,4; 5,6")
>>> B
matrix([[ 1.,  2.],
        [ 3.,  4.],
        [ 5.,  6.]])
```


Python tutorial

A vector as a matrix

Vectors are handled as matrices with one row or one column:

```
>>> x = np.matrix("10., 20.")
>>> x
matrix([[ 10.,  20.]])
>>> x.T
matrix([[ 10.],
        [ 20.]])
```

Here is an example for matrix and vector multiplication:

```
>>> x = np.matrix("4.;5.")
>>> x
matrix([[ 4.],
        [ 5.]])

>>> A = np.matrix([[1.,2], [3,4], [5,6]])
>>> A
matrix([[ 1.,  2.],
        [ 3.,  4.],
        [ 5.,  6.]])

>>> A*x
matrix([[ 14.],
        [ 32.],
        [ 50.]])
```

For vectors, indexing requires two indices:

```
>>> print x[0,0], x[1,0]
4.0 5.0
```

Note

Though np.matrix takes a real matrix form and look pleasing, usually, for most of the cases, arrays are good enough.

Rank

Strings - Escape Sequence, Raw String, and Slicing (/python/python_strings.php)

Strings - Methods (/python/python_strings_method

Formatting Strings - expressions and method calls (/python/python_string_formattir

Files and os.path (/python/python_files.php)

Traversing directories recursively (/python/python_traversing_dire

Subprocess Module (/python/python_subprocess_mc

Regular Expressions with Python (/python/python_regularExpressi

Object Types - Lists (/python/python_lists.php)

Object Types - Dictionaries and Tuples (/python/python_dictionaries_tup

Functions def, *args, **kargs (/python/python_functions_def.p

Functions lambda (/python/python_functions_lamb

Built-in Functions (/python/python_functions_built_

map, filter, and reduce (/python/python_fncs_map_filter

Decorators (/python/python_decorators.php

List Comprehension (/python/python_list_comprehen

Sets (union/intersection) and itertools - Jaccard coefficient and shingling to check

```
>>> import numpy as np
>>> A = np.ones((4,3))
>>> A
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
>>> np.rank(A)
2
```

Note that the rank of the array is not the **rank** of the matrix in linear algebra (dimension of the column space) but **the number of subscripts** it takes!

Scalars have rank **0**:

```
>>> x = np.array(10)
>>> x
array(10)
>>> np.rank(x)
0
```

NumPy supports arrays of any dimension such as rank 3 (2x2x2):

```
>>> A = np.ones((2,2,2))
>>> A
array([[[ 1.,  1.],
        [ 1.,  1.]],

       [[ 1.,  1.],
        [ 1.,  1.]]])
>>> A[1,0,1]
1.0
```

dot product

```
>>> A = np.array([[1,2],[3,4]])
>>> A
array([[1, 2],
       [3, 4]])
>>> b = np.array([10, 20])
>>> b
array([10, 20])
>>> ans = np.dot(A,b)
>>> ans
array([ 50, 110])
```

Another example:

plagiarism
(/python/python_sets_union_inte

Hashing (Hash tables and
hashlib)
(/python/python_hash_tables_ha

Dictionary Comprehension with
zip
(/python/python_dictionary_com

The yield keyword
(/python/python_function_with_y

Generator Functions and
Expressions
(/python/python_generators.php

generator.send() method
(/python/python_function_with_g

Iterators
(/python/python_iterators.php)

Classes and Instances (__init__,
__call__, etc.)
(/python/python_classes_instanc

if__name__ == '__main__'
(/python/python_if__name__equa

argparse
(/python/python_argparse.php)

Exceptions
(/python/python_try_except_final

@static method vs class
method
(/python/python_differences_bet

Private attributes and private
methods
(/python/python_private_attribut

bits, bytes, bitstring, and
constBitStream
(/python/python_bits_bytes_bitst

json.dump(s) and json.load(s)
(/python/python-json-dumps-
loads-file-read-write.php)

Python Object Serialization -

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, B = \begin{bmatrix} 7 & 8 & 9 \end{bmatrix}$$

```
>>> A = np.array([[ 1, 2 ,3], [ 4, 5 ,6]])
>>> B = np.array([7,8,9])
>>>
>>> A
array([[1, 2, 3],
       [4, 5, 6]])
>>>
>>> B
array([7, 8, 9])
>>>
>>> A.shape
(2, 3)
>>> B.shape
(3,)
>>>
>>> A.T
array([[1, 4],
       [2, 5],
       [3, 6]])
>>>
>>> B.T
array([7, 8, 9])
>>>
>>> A.dot(B)
array([ 50, 122])
>>>
>>> np.dot(A,B)
array([ 50, 122])
```

Ax = b : numpy.linalg

Now we want to solve **Ax = b**:

```
>>> import numpy as np
>>> from numpy.linalg import solve
>>> A = np.array([[1,2],[3,4]])
>>> A
array([[1, 2],
       [3, 4]])
>>> b = np.array([10, 20])
>>> b
>>> x = solve(A,b)
>>> x
array([ 0.,  5.])
```

eigen values and vectors

pickle and json
(/python/python_serialization_pic

Python Object Serialization - yaml and json
(/python/python_yaml_json_conv

Priority queue and heap queue data structure
(/python/python_PriorityQueue_l

Graph data structure
(/python/python_graph_data_stru

Dijkstra's shortest path algorithm
(/python/python_Dijkstras_Short

Prim's spanning tree algorithm
(/python/python_Prims_Spanning

Closure
(/python/python_closure.php)

Functional programming in Python
(/python/python_functional_prog

Remote running a local file using ssh
(/python/python_ssh_remote_ru

SQLite 3 - A. Connecting to DB, create/drop table, and insert data into a table
(/python/python_sqlite_connect_

SQLite 3 - B. Selecting, updating and deleting data
(/python/python_sqlite_select_up

MongoDB with PyMongo I - Installing MongoDB ...
(/python/MongoDB_PyMongo/py

Python HTTP Web Services - urllib, httplib2
(/python/python_http_web_servi

Web scraping with Selenium for checking domain availability
(/python/python_Web_scraping_

REST API : Http Requests for

```
>>> import numpy as np
>>> from numpy.linalg import eig
>>> A = np.array([[1,2],[3,4]])
>>> eig(A)
(array([-0.37228132,  5.37228132]), array([[-0.82456484, -0.41597356],
       [ 0.56576746, -0.90937671]]))
```

The **eig** returns two tuples: the first one is the eigen values and the second one is a matrix whose columns are the two eigen vectors.

We can unpack the tuples:

```
>>> eigen_val, eigen_vec = eig(A)
>>> eigen_val
array([-0.37228132,  5.37228132])
>>> eigen_vec
array([[-0.82456484, -0.41597356],
       [ 0.56576746, -0.90937671]])
```

Quadrature

We want to solve $\int_0^3 x^4 dx = \frac{243}{4}$:

```
>>> from scipy.integrate import quad
>>> def f(x):
...     return x**4
...
>>> quad(f, 0., 3.)
(48.599999999999994, 5.39568389967826e-13)
```

The returned tuple indicates (ans, error estimate).

We can get the same answer if we use **lambda** instead:

```
>>> quad(lambda x: x**4, 0, 3)
(48.599999999999994, 5.39568389967826e-13)
```

Python tutorial

Humans with Flask
(/python/python-REST-API-Http-Requests-for-Humans-with-Flask.php)

Blog app with Tornado
(/python/Tornado/Python_Torna

Multithreading ...
(/python/Multithread/python_mu

Python Network Programming I - Basic Server / Client : A Basics
(/python/python_network_progra

Python Network Programming I - Basic Server / Client : B File Transfer
(/python/python_network_progra

Python Network Programming II - Chat Server / Client
(/python/python_network_progra

Python Network Programming III - Echo Server using socketserver network framework
(/python/python_network_progra

Python Network Programming IV - Asynchronous Request Handling : ThreadingMixIn and ForkingMixIn
(/python/python_network_progra

Python Interview Questions I
(/python/python_interview_quest

Python Interview Questions II
(/python/python_interview_quest

Python Interview Questions III
(/python/python_interview_quest

Python Interview Questions IV
(/python/python_interview_quest

Python Interview Questions V
(/python/python_interview_quest

Python Interview Questions VI
(/python/python_interview_quest

Celluar Automata (/python/python_cellular_automata.php)

Batch gradient descent algorithm (/python/python_numpy_batch_gradient_descent_algorithm.php)

Longest Common Substring Algorithm (/python/python_longest_common_substring_lcs_algorithm_generalized_suffix_tree.php)

Python Unit Test - TDD using unittest.TestCase class (/python/python_unit_testing.php)

Simple tool - Google page ranking by keywords (/python/python_site_page_ranking_by_keywords.php)

Google App Hello World (/python/GoogleApp/python_GoogleApp_HelloWorld.php)

Google App webapp2 and WSGI (/python/GoogleApp/python_GoogleApp_WebApp2_WSGI.php)

Uploading Google App Hello World (/python/GoogleApp/python_GoogleApp_Uploading_HelloWorld.php)

Python 2 vs Python 3 (/python/python_differences_Python2_vs_Python3_port.php)

virtualenv and virtualenvwrapper (/python/python_virtualenv_virtualenvwrapper.php)

Uploading a big file to AWS S3 using boto module (/DevOps/AWS/aws_S3_uploading_large_file.php)

Scheduled stopping and starting an AWS instance (/DevOps/AWS/aws_stopping_starting_instances.php)

Cloudera CDH5 - Scheduled stopping and starting services (/Hadoop/BigData_hadoop_CDH5_stop_start_services.php)

Removing Cloud Files - Rackspace API with curl and subprocess (/python/python_Rackspace_API_curl_subprocess_Cloud_Files.php)

Checking if a process is running/hanging and stop/run a scheduled task on Windows (/python/python-Windows-Check-if-a-Process-is-Running-Hanging-Schtasks-Run-Stop.php)

Apache Spark 1.3 with PySpark (Spark Python API) Shell (/Hadoop/BigData_hadoop_Apache_Spark_PySpark.php)

Apache Spark 1.2 Streaming (/Hadoop/BigData_hadoop_Apache_Spark_Streaming.php)

bottle 0.12.7 - Fast and simple WSGI-micro framework for small web-applications ... (/python/Bottle/Python_Bottle_Framework.php)

Flask app with Apache WSGI on Ubuntu14/CentOS7 ... (/python/Flask/Python_Flask_Blog_App_Production_with_MongoDB_and_Apache_WSGI.php)

Fabric - streamlining the use of SSH for application deployment (/python/Fabric/python_Fabric.php)

Ansible Quick Preview - Setting up web servers with Nginx, configure enviroments, and deploy an App
(/DevOps/Ansible/Ansible_SettingUp_Webservers_Nginx_Install_Env_Configure_Deploy_App.php)

Django 1.6/1.8 Web Framework ...
(/python/Django/Python_Django_

Sponsor Open Source development activities and free contents for everyone.

Thank you.

- K Hong (http://bogotobogo.com/about_us.php)

OpenCV 3 image and video processing with Python

Neural Networks with backpropagation for XOR using one hidden layer (/python/python_Neural_Networks_Backpropagation_for_XOR_using_one_hidden_layer.php)

NLP - NLTK (Natural Language Toolkit) ... (/python/NLTK/NLTK_install.php)

RabbitMQ(Message broker server) and Celery(Task queue) ... (/python/RabbitMQ_Celery/python_Installing_RabbitMQ_Celery.php)

OpenCV3 and Matplotlib ... (/python/OpenCV_Python/python_opencv3_matplotlib_rgb_brg_image_load_display_save.php)

Simple tool - Concatenating slides using FFmpeg ... (/FFMpeg/ffmpeg_fade_in_fade_out_transitions_effects_filters_slideshow_concat.php)

iPython - Signal Processing with NumPy (/python/OpenCV_Python/python_opencv3_NumPy_Arrays_Signal_Processing_iPython.php)

iPython and Jupyter - Install Jupyter, iPython Notebook, drawing with Matplotlib, and publishing it to Github (/python/IPython/IPython_Jupyter_Install_iPython_Notebook_Matplotlib_Publishing_it_to_Github.php)

iPython and Jupyter Notebook with Embedded D3.js (/python/IPython/iPython_Jupyter_Notebook_with_Embedded_D3.php)

Downloading YouTube videos using youtube-dl embedded with Python (/VideoStreaming/YouTube/youtube-dl-embedding.php)

Machine Learning : scikit-learn ... (/python/scikit-learn/scikit_machine_learning_Supervised_Learning_Unsupervised_Learning.php)

Django 1.6/1.8 Web Framework ... (/python/Django/Python_Django_tutorial_introduction.php)

OpenCV 3 with Python (/python/OpenCV_Python/pythor

Image - OpenCV BGR : Matplotlib RGB (/python/OpenCV_Python/pythor

Basic image operations - pixel access (/python/OpenCV_Python/pythor

iPython - Signal Processing with NumPy (/python/OpenCV_Python/pythor

Signal Processing with NumPy I - FFT and DFT for sine, square waves, unitpulse, and random signal (/python/OpenCV_Python/pythor

Signal Processing with NumPy II - Image Fourier Transform : FFT & DFT (/python/OpenCV_Python/pythor

Inverse Fourier Transform of an Image with low pass filter: cv2.idft() (/python/OpenCV_Python/pythor

Image Histogram (/python/OpenCV_Python/pythor

Video Capture and Switching colorspaces - RGB / HSV (/python/OpenCV_Python/pythor

Adaptive Thresholding - Otsu's clustering-based image thresholding (/python/OpenCV_Python/pythor

Edge Detection - Sobel and Laplacian Kernels (/python/OpenCV_Python/pythor

Canny Edge Detection (/python/OpenCV_Python/pythor

Hough Transform - Circles (/python/OpenCV_Python/pythor

Watershed Algorithm : Marker-based Segmentation I
(/python/OpenCV_Python/pythor

Watershed Algorithm : Marker-based Segmentation II
(/python/OpenCV_Python/pythor

Image noise reduction : Non-local Means denoising algorithm
(/python/OpenCV_Python/pythor
local_Means_Denoising_Algorithr

Image object detection : Face detection using Haar Cascade Classifiers
(/python/OpenCV_Python/pythor

Image segmentation - Foreground extraction Grabcut algorithm based on graph cuts
(/python/OpenCV_Python/pythor

Image Reconstruction - Inpainting (Interpolation) - Fast Marching Methods
(/python/OpenCV_Python/pythor

Video : Mean shift object tracking
(/python/OpenCV_Python/pythor

Machine Learning : Clustering - K-Means clustering I
(/python/OpenCV_Python/pythor
Means_Clustering_Vector_Quanti

Machine Learning : Clustering - K-Means clustering II
(/python/OpenCV_Python/pythor
Means_Clustering_Vector_Quanti

Machine Learning : Classification - k-nearest neighbors (k-NN) algorithm
(/python/OpenCV_Python/pythor
nearest_neighbors_k-NN.php)

Machine Learning with scikit-learn

MACHINE LEARNING ALGORITHMS AND CONCEPTS

ARTIFICIAL NEURAL NETWORKS (ANN)

[Note] Sources are available at Github - Jupyter notebook files (https://github.com/Einsteinish/A Neural-Networks-with-Jupyter.git)

ANN-9-Deep-Learning-2-Image-Recognition-Image-Classification.php)

10 - Deep Learning III : Deep Learning III : Theano, TensorFlow, and Keras (/python/scikit-learn/Artificial-Neural-Network-ANN-10-Deep-Learning-3-Theano-TensorFlow-Keras.php)

CONTACT

BogoToBogo
contactus@bogotobogo.com (mailto:#)

FOLLOW BOGOTOBOGO

(https://www.facebook.com/KHongSanFrancisco) (https://twitter.com/KHongTwit) (https://plus.google.com/u/0/+KHongSanFrancisco/posts)

ABOUT US (/ABOUT_US.PHP)

contactus@bogotobogo.com (mailto:#)

Golden Gate Ave, San Francisco, CA 94115

Golden Gate Ave, San Francisco, CA 94115