

NumPy

NumPy (pronounced /ˈnʌmpaɪ/ (*NUM-py*) or sometimes /ˈnʌmpi/^[1]^[2] (*NUM-pee*)) is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

Contents

History

Traits

- The ndarray data structure
- Limitations

Examples

See also

References

Further reading

External links

History

The Python programming language was not initially designed for numerical computing, but attracted the attention of the scientific and engineering community early on, so that a special interest group called matrix-sig was founded in 1995 with the aim of defining an array computing package. Among its members was Python designer and maintainer Guido van Rossum, who implemented extensions to Python's syntax (in particular the indexing syntax) to make array computing easier.^[3]

NumPy



Original author(s)	Travis Oliphant
Developer(s)	Community project
Initial release	As Numeric, 1995; as NumPy, 2006
Stable release	1.15.1 / 21 August 2018
Repository	https://github.com/numpy/numpy
Written in	Python, C
Operating system	Cross-platform
Type	Technical computing
License	BSD-new license
Website	www.numpy.org (https://www.numpy.org/)

An implementation of a matrix package was completed by Jim Fulton, then generalized by [Jim Hugunin](#) to become *Numeric*,^[3] also variously called Numerical Python extensions or NumPy.^{[4][5]} Hugunin, a graduate student at MIT,^{[5]:10} joined [CNRI](#) to work on [JPython](#) in 1997^[3] leaving Paul Dubois of [LLNL](#) to take over as maintainer.^{[5]:10} Other early contributors include David Ascher, Konrad Hinsen and [Travis Oliphant](#).^{[5]:10}

A new package called *Numarray* was written as a more flexible replacement for Numeric.^[6] Like Numeric, it is now deprecated.^{[7][8]} Numarray had faster operations for large arrays, but was slower than Numeric on small ones,^[9] so for a time both packages were used for different use cases. The last version of Numeric v24.2 was released on 11 November 2005 and numarray v1.5.2 was released on 24 August 2006.^[10]

There was a desire to get Numeric into the Python standard library, but Guido van Rossum decided that the code was not maintainable in its state then.^[11]

In early 2005, NumPy developer [Travis Oliphant](#) wanted to unify the community around a single array package and ported Numarray's features to Numeric, releasing the result as NumPy 1.0 in 2006.^[6] This new project was part of [SciPy](#). To avoid installing the large SciPy package just to get an array object, this new package was separated and called NumPy. Support for Python 3 was added in 2011 with NumPy version 1.5.0.^[12]

In 2011, [PyPy](#) started development on an implementation of the NumPy API for PyPy.^[13] It is not yet fully compatible with NumPy.^[14]

Traits

NumPy targets the [CPython](#) reference implementation of Python, which is a non-optimizing [bytecode](#) interpreter. Mathematical algorithms written for this version of Python often run much slower than [compiled](#) equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops using NumPy.

Using NumPy in Python gives functionality comparable to [MATLAB](#) since they are both interpreted,^[15] and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of [scalars](#). In comparison, MATLAB boasts a large number of additional toolboxes, notably [Simulink](#), whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; [SciPy](#) is a library that adds more MATLAB-like functionality and [Matplotlib](#) is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on [BLAS](#) and [LAPACK](#) for efficient linear algebra computations.

Python [bindings](#) of the widely used [computer vision](#) library [OpenCV](#) utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, [slicing](#) or [masking](#) with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted [feature points](#), [filter kernels](#) and many more vastly simplifies the programming workflow and [debugging](#).

The ndarray data structure

The core functionality of NumPy is its "ndarray", for n -dimensional array, data structure. These arrays are strided views on memory.^[6] In contrast to Python's built-in list data structure (which, despite the name, is a dynamic array), these arrays are homogeneously typed: all elements of a single array must be of the same type.

Such arrays can also be views into memory buffers allocated by C/C++, Cython, and Fortran extensions to the CPython interpreter without the need to copy data around, giving a degree of compatibility with existing numerical libraries. This functionality is exploited by the SciPy package, which wraps a number of such libraries (notably BLAS and LAPACK). NumPy has built-in support for memory-mapped ndarrays.^[6]

Limitations

Inserting or appending entries to an array is not as trivially possible as it is with Python's lists. The `np.pad(...)` routine to extend arrays actually creates new arrays of the desired shape and padding values, copies the given array into the new one and returns it. NumPy's `np.concatenate([a1,a2])` operation does not actually link the two arrays but returns a new one, filled with the entries from both given arrays in sequence. Reshaping the dimensionality of an array with `np.reshape(...)` is only possible as long as the number of elements in the array does not change. These circumstances originate from the fact that NumPy's arrays must be views on contiguous memory buffers. A replacement package called Blaze attempts to overcome this limitation.^[16]

Algorithms that are not expressible as a vectorized operation will typically run slowly because they must be implemented in "pure Python", while vectorization may increase memory complexity of some operations from constant to linear, because temporary arrays must be created that are as large as the inputs. Runtime compilation of numerical code has been implemented by several groups to avoid these problems; open source solutions that interoperate with NumPy include `scipy.weave`, `numexpr`^[17] and `Numba`.^[18] Cython is a static-compiling alternative to these.

Examples

Array creation

```
>>> import numpy as np
>>> x = np.array([1, 2, 3])
>>> x
array([1, 2, 3])
>>> y = np.arange(10) # Like Python's range, but returns an array
>>> y
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Basic operations

```
>>> a = np.array([1, 2, 3, 6])
>>> b = np.linspace(0, 2, 4) # create an array with four equally spaced points starting with 0 and ending with 2.
>>> c = a - b
>>> c
```

```
array([ 1.          ,  1.33333333,  1.66666667,  4.          ])
>>> a**2
array([ 1,  4,  9, 36])
```

Universal functions

```
>>> a = np.linspace(-np.pi, np.pi, 100)
>>> b = np.sin(a)
>>> c = np.cos(a)
```

Linear algebra

```
>>> from numpy.random import rand
>>> from numpy.linalg import solve, inv
>>> a = np.array([[1, 2, 3], [3, 4, 6.7], [5, 9.0, 5]])
>>> a.transpose()
array([[ 1. ,  3. ,  5. ],
       [ 2. ,  4. ,  9. ],
       [ 3. ,  6.7,  5. ]])
>>> inv(a)
array([[-2.27683616,  0.96045198,  0.07909605],
       [ 1.04519774, -0.56497175,  0.1299435 ],
       [ 0.39548023,  0.05649718, -0.11299435]])
>>> b = np.array([3, 2, 1])
>>> solve(a, b) # solve the equation ax = b
array([-4.83050847,  2.13559322,  1.18644068])
>>> c = rand(3, 3) * 20 # create a 3x3 random matrix of values within [0,1] scaled by 20
>>> c
array([[ 3.98732789,  2.47702609,  4.71167924],
       [ 9.24410671,  5.5240412 , 10.6468792 ],
       [10.38136661,  8.44968437, 15.17639591]])
>>> np.dot(a, c) # matrix multiplication
array([[ 53.61964114,  38.8741616 ,  71.53462537],
       [118.4935668 ,  86.14012835, 158.40440712],
       [155.04043289, 104.3499231 , 195.26228855]])
>>> a @ c # Starting with Python 3.5 and NumPy 1.10
array([[ 53.61964114,  38.8741616 ,  71.53462537],
       [118.4935668 ,  86.14012835, 158.40440712],
       [155.04043289, 104.3499231 , 195.26228855]])
```

Incorporation with OpenCV

```
>>> import numpy as np
>>> import cv2
>>> r = np.reshape(np.arange(256*256)%256,(256,256)) # 256x256 pixel array with a horizontal gradient from 0 to 255 for the red color channel
>>> g = np.zeros_like(r) # array of same size and type as r but filled with 0s for the green color channel
>>> b = r.T # transposed r will give a vertical gradient for the blue color channel
>>> cv2.imwrite('gradients.png', np.dstack([b,g,r])) # OpenCV images are interpreted as BGR, the depth-stacked array will be written to an 8bit RGB PNG-file
```

```
called 'gradients.png'
True
```

Nearest Neighbor Search - Iterative Python algorithm and vectorized NumPy version

```
>>> ### Pure iterative Python ###
>>> points = [[9,2,8],[4,7,2],[3,4,4],[5,6,9],[5,0,7],[8,2,7],[0,3,2],[7,3,0],[6,1,1],[2,9,6]]
>>> qPoint = [4,5,3]
>>> minIdx = -1
>>> minDist = -1
>>> for idx, point in enumerate(points): # iterate over all points
>>>     dist = sum([(dp-dq)**2 for dp,dq in zip(point,qPoint)])**0.5 # compute the euclidean distance for each point to q
>>>     if dist < minDist or minDist < 0: # if necessary, update minimum distance and index of the corresponding point
>>>         minDist = dist
>>>         minIdx = idx
>>> print 'Nearest point to q: ', points[minIdx]
Nearest point to q:  [3, 4, 4]


>>> ### Equivalent NumPy vectorization ###
>>> import numpy as np
>>> points = np.array([[9,2,8],[4,7,2],[3,4,4],[5,6,9],[5,0,7],[8,2,7],[0,3,2],[7,3,0],[6,1,1],[2,9,6]])
>>> qPoint = np.array([4,5,3])
>>> minIdx = np.argmin(np.linalg.norm(points-qPoint,axis=1)) # compute all euclidean distances at once and return the index of the smallest one
>>> print 'Nearest point to q: ', points[minIdx]
Nearest point to q:  [3 4 4]
```

See also

- List of numerical analysis software
- SciPy
- matplotlib

References

- Pine, David (2014). "Python resources" (http://www.physics.nyu.edu/pine/pymanual/html/apdx3/apdx3_resources.html). Rutgers University. Retrieved 2017-04-07.
- "How do you say numpy?" (https://www.reddit.com/r/Python/comments/2709pq/how_do_you_say_numpy/). Reddit. 2015. Retrieved 2017-04-07.
- Millman, K. Jarrod; Aivazis, Michael (2011). "Python for Scientists and Engineers" (<http://www.computer.org/csdl/mags/cs/2011/02/mcs2011020009.html>). *Computing in Science and Engineering*. **13** (2): 9–12.
- Travis Oliphant (2007). "Python for Scientific Computing" (http://www.vision.ime.usp.br/~thsant/pool/oliphant-python_scientific.pdf) (PDF). *Computing in Science and Engineering*.

5. David Ascher; Paul F. Dubois; Konrad Hinsen; Jim Hugunin; Travis Oliphant (1999). "Numerical Python" (<http://www.cs.mcgill.ca/~hv/articles/Numerical/numpy.pdf>) (PDF).
6. van der Walt, Stéfan; Colbert, S. Chris; Varoquaux, Gaël (2011). "The NumPy array: a structure for efficient numerical computation". *Computing in Science and Engineering*. IEEE. [arXiv:1102.1523](https://arxiv.org/abs/1102.1523) (<https://arxiv.org/abs/1102.1523>)  [Bibcode:2011arXiv1102.1523V](https://arxiv.org/abs/1102.1523) (<http://adsabs.harvard.edu/abs/2011arXiv1102.1523V>).
7. "Numarray Homepage" (http://www.stsci.edu/resources/software_hardware/numarray). Retrieved 2006-06-24.
8. Travis E. Oliphant (7 December 2006). *Guide to NumPy* (<https://archive.org/details/NumPyBook>). Retrieved 2 February 2017.
9. Travis Oliphant and other SciPy developers. "[Numpy-discussion] Status of Numeric" (<https://mail.scipy.org/pipermail/numpy-discussion/2004-January/002645.html>). Retrieved 2 February 2017.
10. "NumPy Sourceforge Files" (http://sourceforge.net/project/showfiles.php?group_id=1369). Retrieved 2008-03-24.
11. "History_of_SciPy - SciPy wiki dump" (https://scipy.github.io/old-wiki/pages/History_of_SciPy.html). *scipy.github.io*.
12. "NumPy 1.5.0 Release Notes" (<http://sourceforge.net/projects/numpy/files/NumPy/1.5.0/NOTES.txt/view>). Retrieved 2011-04-29.
13. "PyPy Status Blog: NumPy funding and status update" (<http://morepypy.blogspot.com/2011/10/numpy-funding-and-status-update.html>). Retrieved 2011-12-22.
14. "NumPyPy Status" (<http://buildbot.pypy.org/numpy-status/latest.html>). Retrieved 2013-10-14.
15. The SciPy Community. "NumPy for Matlab users" (<https://docs.scipy.org/doc/numpy-dev/user/numpy-for-matlab-users.html>). Retrieved 2 February 2017.
16. "Blaze Ecosystem Docs" (<https://blaze.readthedocs.io/>). *Read the Docs*. Retrieved 17 July 2016.
17. Francesc Alted. "numexpr" (<https://github.com/pydata/numexpr>). Retrieved 8 March 2014.
18. "Numba" (<http://numba.pydata.org/>). Retrieved 8 March 2014.

Further reading

- Bressert, Eli (2012). *Scipy and Numpy: An Overview for Developers*. O'Reilly Media. ISBN 978-1-4493-0546-8.
- McKinney, Wes (2017). *Python for Data Analysis : Data Wrangling with Pandas, NumPy, and IPython* (2nd ed.). Sebastopol: O'Reilly. ISBN 978-1-4919-5766-0.

External links

- Official website (<https://www.numpy.org/>)
 - History of NumPy (https://scipy.github.io/old-wiki/pages/History_of_SciPy)
-

Retrieved from "<https://en.wikipedia.org/w/index.php?title=NumPy&oldid=857157384>"

This page was last edited on 29 August 2018, at 22:53 (UTC).

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.