Numpy

Christopher Barker

IRIS

October 23, 2013

Table of Contents

numpy

numpy

Not just for lots of numbers!

(but it's great for that!)

`http://www.numpy.org/`

what is numpy?

An N-Dimensional array object

A whole pile of tools for operations on/with that object.

Why numpy?

Classic answer: Lots of numbers

- Faster
- Less memory
- More data types

Even if you don't have lot of numbers:

- N-d array slicing
- Vector operations
- Flexible data types

why numpy?

Wrapper for a block of memory:

- Interfacing with C libs
- PyOpenGL
- GDAL
- NetCDF4
- Shapely

Image processing:

- PIL
- WxImage
- ndimage

What is an nd array?

- N-dimensional (up to 32!)
- Homogeneous array:
  - Every element is the same type
    (but that type can be a pyObject)
  - Int, float, char – more exotic types
- "rank" number of dimensions
- Strided data:
  - Describes how to index into block of memory
  - PEP 3118 – Revising the buffer protocol

demos: `memory.py` and `structure.py`

Built-in Data Types

- Signed and unsigned Integers
  8, 16, 32, 64 bits
- Floating Point
  32, 64, 96, 128 bits (not all platforms)
- Complex
  64, 128, 192, 256 bits
- String and unicode
  Static length
- Bool
  8 bit
- Python Object
  Really a pointer

demo: `object.py`

Compund dtypes

- Can define any combination of other types
  Still Homogenous: Array of structs.
- Can name the fields
- Can be like a database table
- Useful for reading binary data

demo: `dtypes.py`

Array Constructors:

From scratch:

`ones()`, `zeros()`, `empty()`, `arange()`, `linspace()`, `logspace()`

(Default dtype: np.float64)

From sequences:

`array()`, `asarray()`

(Build from any sequence)

From binary data:

`fromstring()`, `frombuffer()`, `fromfile()`

Assorted linear algebra standards:

`eye()`, `diag()`, `etc.`

demo: `constructors.py`

Broadcasting:

Element-wise operations among two different rank arrays:

Simple case: scalar and array:

```
In [37]: a
Out[37]: array([1, 2, 3])

In [38]: a*3
Out[38]: array([3, 6, 9])
```

Great for functions of more than one variable on a grid

demo: `broadcasting.py`

Slicing – views:

a slice is a "view" on the array:
new object, but shares memory:

```
In [12]: a = np.array((1,2,3,4))
In [13]: b = a[:]
# for lists -- [:] means copy -- not for arrays!
In [15]: a is b
Out[15]: False
# it's new array, but...
In [16]: b[2] = 5
In [17]: a
Out[17]: array([1, 2, 5, 4])
# a and b share data
```

demo: `slice.py`

Working with compiled code:

Wrapper around a C pointer to a block of data

- Some code can't be vectorized
- Interface with existing libraries

Tools:

- C API: you don't want to do that!
- Cython: typed arrays
- Ctypes
- SWIG: numpy.i
- Boost: boost array
- f2py

Example of numpy+cython:
`http://wiki.cython.org/examples/mandelbrot`

numpy persistance:

`.tofile() / fromfile()`

– Just the raw bytes, no metadata

pickle

`savez()` – numpy zip format

Compact: binary dump plus metadata

netcdf

Hdf

- Pyhdf
- pytables

Other stuff:

- Masked arrays
- Memory-mapped files
- Set operations: unique, etc
- Random numbers
- Polynomials
- FFT
- Sorting and searching
- Linear Algebra
- Statistics

(And all of scipy!)

numpy docs:

`www.numpy.org`
– Numpy reference Downloads, etc

`www.scipy.org`
– lots of docs

Scipy cookbook
`http://www.scipy.org/Cookbook`

“The Numpy Book”
`http://www.tramy.us/numpybook.pdf`