NumPy & SciPy Fundamentals

Learning CS with Python Series - Day 3

What is NumPy

- It is a **class** that provides numerical functions
  - This includes:
    - Matrix support
      - Summery Information (shape, sum, max, min, mean, etc)
    - Dot products
    - Inverting and Transposing

Recap of Types

- As discussed on day one, there is only really binary ints
  - But we can “fake” strings by grouping some of those bits into bytes
  - And we can “fake” floating point numbers by grouping the bits into a sign, exponent and fraction
- Lists and Dictionaries are just groups of the above types
  - Now let’s discuss NumPy arrays

NumPy arrays ... They aren't that special

- They aren't magic
- They are just grouping lists and constraining the individual elements to numeric types
- So it's a layered cake:
  - floats are built on integers
  - lists are built by grouping floats
  - NumPy arrays are groups of lists

Example

```
import numpy

a = numpy.array([
    [0, 1, 2],
    [3, 4, 5],
    [6, 7, 8]
])
```

```
a.max() # 8

a.max(axis=0) # array([6, 7, 8])

a.mean() # 4

a.mean(axis=1) # array([1, 4, 7])

a[1, 1] # 4
```

Import CSV Data

```
import csv
import numpy
...
myfilepointer = csv.reader(open(filename,'rb'))
for row in myfilepointer:
    data.append(row)
```

Import CSV Data

```
import numpy
...
data = numpy.genfromtxt(filename,delimiter=’,’)
```

Performing Linear Algebra

```
import numpy

...

c = numpy.dot(a,b)

d = c.transpose()

i = numpy.linalg.inv(a)
```

OLS Class

```
import numpy

class MyOLS:

    def __init(self, x, y)__:

        self.x = x

        self.y = y

        self.tx = x.T # shortcut to .transpose()

        self.xx = numpy.dot(x,tx)

        self.xxi = numpy.linalg.inv(xx)

    def EstimateBeta(self):

        return numpy.dot(numpy.dot(self.xxi,self.tx),self.y)
```

OLS Class

```
import numpy

class MyOLS:

...

    def wald(self, R, c):
    b = self.EstimateBeta()

    ymxb = self.y-numpy.dot(self.x,b)

    e = numpy.dot(ymxb.T, ymxb)/(self.x.shape[0]-self.x.shape[1])

    sides = numpy.dot(R,b)-c

    center = e*numpy.dot(numpy.dot(R,self.xxi),R.T)

    return numpy.dot(numpy.dot(sides.T,numpy.linalg.inv(center)),sides)
```

So what's in SciPy

- It's just a class
  - It just happens to built on NumPy
- Has many convenient tools for scientific computing
  - Pulling numbers from distributions
  - Optimizations
  - Regressions

Pulling Numbers

```
import scipy
...
scipy.stats.norm.rvs(2, 3)
```

Optimizations

```
import scipy.optimize as opt

myfunc = lambda (x,y): (1-x)**2 + 100*(y-x**2)**2

myder = lamda (x): array([2*100*(x[1] - x[0]**2)*(-2*x[0]) - 2*(1.-x[0]), 2*100*(x[1]-
x[0]**2)])

(bfgsx,bfgsy) = opt.fmin_bfgs(myfunc, (2,2),myder)
```

Regression

```
from scipy import stats

...

(slope, intercept, r_value, p_value, std_err) = stats.linregress(x,y)
```