SciPy

Basic functions

Interaction with Numpy

Scipy builds on Numpy, and for all basic array handling needs you can use Numpy functions:

```
import numpy as np
np.some_function()
```

To use functions from some of the Scipy modules, you can do

```
from scipy import some_module
some_module.some_function()
```

Index Tricks

```
operation of np.mgrid , np.ogrid , np.r_ , and np.c_ for quickly constructing
arrays.

rather than writing something like the following
a = np.concatenate(([3], [0]*5, np.arange(-1, 1.002, 2/9.0)))
```

with the `r_` command (row concatination) one can enter this as

```
a = np.r_[3,[0]*5,-1:1:10j]
```

c_ that stacks 2d arrays by columns but works identically to r_ for 1d arrays

mgrid and ogrid

```
np.mgrid[0:5,0:5]
```

Vectorizing functions (vectorize)

suppose you have a Python function named `addsubtract` defined as

```
        def addsubtract(a,b):
        ...     if a > b:
        ...         return a - b
        ...     else:
        ...         return a + b
```

he class vectorize can be used to “vectorize “ this function so that

```
        vec_addsubtract = np.vectorize(addsubtract)
```

returns a function which takes array arguments and returns an array result

```
        vec_addsubtract([0,3,6,9],[1,3,5,7])
```

Type handling

np.cast[‘f ’](np.pi)

Other useful functions

linspace and logspace - return equally spaced samples in a linear or log scale

select - select which extends the functionality of where to include multiple conditions and multiple choices. select is a vectorized form of the multiple if-statement.

```
x = np.r_[-2:3]
print(x)
array([-2, -1,  0,  1,  2])
y = np.select([x > 3, x >= 0], [0, x+2])
print(y)
array([0, 0, 2, 3, 4])
```

scipy.stats

```
info(stats)

dir(norm)

from scipy import stats
from scipy.stats import norm
```

Common Methods

- rvs: Random Variates
- pdf: Probability Density Function
- cdf: Cumulative Distribution Function
- sf: Survival Function (1-CDF)
- ppf: Percent Point Function (Inverse of CDF)
- isf: Inverse Survival Function (Inverse of SF)
- stats: Return mean, variance, (Fisher's) skew, or (Fisher's) kurtosis
- moment: non-central moments of the distribution

```
norm.cdf(0)


norm.cdf([-1., 0, 1])


norm.cdf(np.array([-1., 0, 1]))


norm.mean(), norm.std(), norm.var()


norm.stats(moments="mv")


norm.ppf(0.5)
```

generate a sequence of random variates, use the `size` keyword argument

```
norm.rvs(size=3)
```

`numpy.random` package

To achieve reproducibility, you can explicitly seed a global variable

```
np.random.seed(1234)
```

```
norm.rvs(size=5, random_state=1234)
```

Linear Algebra (`scipy.linalg`)

numpy.matrix vs 2D numpy.ndarray

```
A = np.mat('[1 2;3 4]')

linalg.inv(A)  - inverse of matrix A

linalg.det(A) – Determinant of Matrix A

linalg.norm(A)   - forbenius norm(default),

linalg.norm(A,1) - L1 norm (1),

linalg.norm(A, np.inf)  -  L inf norm(np.inf)
```

`linalg.lstsq` and `linalg.pinv` for solving a data-fitting problem

```
linalg.lstsq – To Calculate Least Square

        c, resid, rank, sigma = linalg.lstsq(A, zi)

linalg.pinv, linalg.pinv2  - To calculate Generalised inverse
```