CS2204

Program Design and Data Structures for Scientific Computing

Announcements

- Project #2 due shortly
- Project #3 will be posted shortly
- Exam #1 very soon

SciPy

- SciPy is a collection of mathematical algorithms
  - Built on top of NumPy
- It exposes the user to high-level commands and classes for the manipulation and visualization of data
- With SciPy, an interactive Python session becomes a data-processing and system-prototyping environment rivaling sytems such as MATLAB, IDL, Octave, R-Lab, and SciLab
  - The additional power of using SciPy within Python, however, is that a powerful programming language is also available for use in developing sophisticated programs

SciPy

- For brevity and convenience, we will often assume that the main packages (numpy, scipy, and matplotlib) have been imported as:

```
>>> import numpy as np
>>> import scipy as sp
>>> import matplotlib as mpl
>>> import matplotlib.pyplot as plt
```

- These are the import conventions used by the Python community
  - You will see these conventions used throughout NumPy and SciPy source code and documentation.
  - You are not required to follow these conventions in your own code, but it is highly recommended

SciPy

- SciPy is organized into subpackages covering different scientific computing domains.
  - This list of subpackages is on the next slide
- Scipy sub-packages need to be imported separately:

  >>> from scipy import linalg, optimize
- Because of their ubiquitousness, some of the functions in these subpackages are also made available in the scipy namespace to ease their use in interactive sessions and programs.
  - In addition, many basic array functions from NumPy are also available at the top-level of the SciPy package.

SciPy

Subpackage	Description
cluster	Clustering algorithms
constants	Physical and mathematical constants
fftpack	Fast Fourier Transform routines
integrate	Integration and ordinary differential equation solvers
interpolate	Interpolation and smoothing splines
io	Input and Output
linalg	Linear algebra
maxentropy	Maximum entropy methods
ndimage	N-dimensional image processing
odr	Orthogonal distance regression
optimize	Optimization and root-finding routines
signal	Signal processing
sparse	Sparse matrices and associated routines
spatial	Spatial data structures and
special	Special functions
stats	Statistical distributions and functions
weave	C/C++ integration

SciPy

- Finding documentation:
  - Available on web at http://docs.scipy.org/
  - Also available via interactive “help” command

```
>>> help()  # start interactive help session
>>> help(linalg)  # get help on a particular package
>>> help(linalg.lstsq)  # get help on a particular function
```

SciPy Vectorize

- Used to convert an ordinary Python function which accepts scalars and returns scalars into a “vectorized-function”
- For example, suppose you have a Python function named `addsubtract` defined as:

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

which defines a function of two scalar variables and returns a scalar result. The class `vectorize` can be used to “vectorize” this function

SciPy Vectorize

- Then

  ```
  >>> vec_addsubtract = vectorize(addsubtract)
  ```

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

  ```
  >>> vec_addsubtract([0,3,6,9],[1,3,5,7]) array([1, 6, 1, 2])
  ```

- This particular function could have been written in vector form without the use of `vectorize` . But, what if the function you have written is the result of some optimization or integration routine. Such functions can likely only be vectorized using `vectorize`.

scipy.integrate

Methods for Integrating Functions given function object.

quad -- General purpose integration.
dblquad -- General purpose double integration.
tplquad -- General purpose triple integration.
fixed_quad -- Integrate func(x) using Gaussian quadrature of order n.
quadrature -- Integrate with given tolerance using Gaussian quadrature.
romberg -- Integrate func using Romberg integration.

Methods for Integrating Functions given fixed samples.

trapz -- Use trapezoidal rule to compute integral from samples.
cumtrapz -- Use trapezoidal rule to cumulatively compute integral.
simps -- Use Simpson's rule to compute integral from samples.
romb -- Use Romberg Integration to compute integral from (2**k + 1) evenly-spaced samples.

Interface to numerical integrators of ODE systems.

odeint -- General integration of ordinary differential equations.
ode -- Integrate ODE using VODE and ZVODE routines.

scipy.optimize

- This module contains:
  1. Unconstrained and constrained minimization and least-squares algorithms (e.g., fmin: Nelder-Mead simplex, fmin_bfgs: BFGS, fmin_ncg: Newton Conjugate Gradient, leastsq: Levenberg-Marquardt, fmin_cobyla: COBYLA).
  2. The unconstrained solvers also have a unified interface, minimize, which can be used to easily compare and switch between algorithms.
  3. Global (brute-force) optimization routines (e.g., anneal)
  4. Curve fitting (curve_fit)
  5. Scalar function minimizers and root finders (e.g., Brent's method fminbound, and newton)
  6. Multivariate equation system solvers (fsolve)
  7. Large-scale multivariate equation system solvers (e.g. newton_krylov)

scipy.interpolate

There are several general interpolation facilities available in SciPy, for data in 1, 2, and higher dimensions:

- A class representing an interpolant (interp1d) in 1-D, offering several interpolation methods.
- Convenience function griddata offering a simple interface to interpolation in N dimensions (N = 1, 2, 3, 4, ...). Object-oriented interface for the underlying routines is also available.
- Functions for 1- and 2-dimensional (smoothed) cubic-spline interpolation, based on the FORTRAN library FITPACK. There are both procedural and object-oriented interfaces for the FITPACK library.
- Interpolation using Radial Basis Functions.

scipy.fftpack

- Fourier analysis is fundamentally a method for expressing a function as a sum of periodic components, and for recovering the signal from those components.
- When both the function and its Fourier transform are replaced with discretized counterparts, it is called the discrete Fourier transform (DFT).
  - The DFT has become a mainstay of numerical computing in part because of a very fast algorithm for computing it, called the Fast Fourier Transform (FFT), which was known to Gauss (1805)
- The package provides: one, two and n dimensional discrete Fourier transforms, and discrete cosine/sine transforms (of different types)

scipy.signal

- The signal processing toolbox currently contains some filtering functions, a limited set of filter design tools, and a few B-spline interpolation algorithms for one- and two-dimensional data.

- While the B-spline algorithms could technically be placed under the interpolation category, they are included here because they only work with equally-spaced data and make heavy use of filter-theory and transfer-function formalism to provide a fast B-spline transform.

- A signal in SciPy is an array of real or complex numbers.

scipy.linalg

- SciPy has very fast linear algebra capabilities
- The linear algebra routines expect an object that can be converted into a 2-dimensional array.
  - The output of these routines is also a two-dimensional array.
- There is a `matrix` class defined in Numpy, which you can initialize with an appropriate Numpy array in order to get objects for which multiplication is matrix-multiplication instead of the default, element-by-element multiplication.
  - You can enter two-dimensional matrices using MATLAB-like syntax with commas or spaces separating columns and semicolons separating rows as long as the matrix is placed in a string passed to `matrix`

scipy.linalg basic routines

- The basic routines of linalg include functions for
  - Finding inverse
  - Solving linear system
  - Finding determinant
  - Computing norms
  - Solving linear least-squares problems

scipy.linalg decompositions

- In many applications it is useful to decompose a matrix using other representations.

- There are several decompositions supported by SciPy:
  - Eigenvalues and eigenvectors
  - Singular value decomposition
  - LU decomposition
  - Cholesky decomposition
  - QR decomposition
  - Schur decomposition

scipy.linalg matrix functions

- Exponential and logarithm functions
- Trigonometric functions
- Hyperbolic trigonometric functions

scipy.linalg matrix creation

Type	Function	Description
block diagonal	scipy.linalg.block_diag	Create a block diagonal matrix from the provided arrays.
circulant	scipy.linalg.circulant	Construct a circulant matrix.
companion	scipy.linalg.companion	Create a companion matrix.
Hadamard	scipy.linalg.hadamard	Construct a Hadamard matrix.
Hankel	scipy.linalg.hankel	Construct a Hankel matrix.
Hilbert	scipy.linalg.hilbert	Construct a Hilbert matrix.
Inverse Hilbert	scipy.linalg.invhilbert	Construct the inverse of a Hilbert matrix.
Leslie	scipy.linalg.leslie	Create a Leslie matrix.
Pascal	scipy.linalg.pascal	Create a Pascal matrix.
Toeplitz	scipy.linalg.toeplitz	Construct a Toeplitz matrix.
Van der Monde	numpy.vander	Generate a Van der Monde matrix.

scipy.stats

- This module contains a large number of probability distributions as well as a growing library of statistical functions.
  - a fairly complete listing of these functions can be had using info(stats).

- There are two general distribution classes that have been implemented for encapsulating continuous random variables and discrete random variables .
  - Over 80 continuous random variables and
  - 10 discrete random variables

Project #3

- We conclude with a discussion of some aspects of Project #3
  - We will discuss reading data from a file
  - And then converting a string of data into a list of data using string operations and list comprehensions