Chapter 4 - Numerical computing

Nicolas P. Rougier

Objectives

The goal of this chapter is to introduce the `numpy` and `scipy` modules that provide a N-dimensional array object and mathematical and numerical associated routines. NumPy (numerical python) provides basic routines for manipulating arrays and matrices while SciPy (scientific python) extends the functionality of NumPy with a substantial collection of common (and not so common) algorithms.

Introduction

Numpy (Oliphant (2006)) is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities
- and a lot more…

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined and this allows NumPy to seamlessly and speedily integrate with a wide variety of projects. We are going to explore numpy through a simple example, implementing the Game of Life.

Broadcasting rules

$$5 \times \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 5 \times 1 & 5 \times 2 \\ 5 \times 3 & 5 \times 4 \end{pmatrix} = \begin{pmatrix} 5 & 10 \\ 15 & 20 \end{pmatrix}$$

```
>>> M = np.array([[1, 2], [3, 4]])
>>> print(5*M)
[[ 5 10]
 [15 20]]
```

Game Of Life

Numpy is slanted toward scientific computing and we'll consider in this section the game of life by John Conway which is one of the earliest example of cellular automata (see figure below). Those cellular automaton can be conveniently considered as array of cells that are connected together through the notion of neighbours. We'll show in the following sections implementation of this game using pure python and numpy in order to illustrate main differences with python and numpy.

```
Z = [[0,0,0,0,0,0],
     [0,0,0,1,0,0],
     [0,1,0,1,0,0],
     [0,0,1,1,0,0],
     [0,0,0,0,0,0],
     [0,0,0,0,0,0]]

def compute_neighbours(Z):
    shape = len(Z), len(Z[0])
    N  = [[0,]*(shape[0])  for i in range(shape[1])]
    for x in range(1,shape[0]-1):
        for y in range(1,shape[1]-1):
            N[x][y] = Z[x-1][y-1]+Z[x][y-1]+Z[x+1][y-1] \
                   + Z[x-1][y]            +Z[x+1][y]   \
                   + Z[x-1][y+1]+Z[x][y+1]+Z[x+1][y+1]
    return N

def show(Z):
    for l in Z[1:-1]:
        print(l[1:-1])
    print()

def iterate(Z):
    shape = len(Z), len(Z[0])
    N = compute_neighbours(Z)
    for x in range(1,shape[0]-1):
        for y in range(1,shape[1]-1):
            if Z[x][y] == 1 and (N[x][y] < 2 or N[x][y] > 3):
                Z[x][y] = 0
            elif Z[x][y] == 0 and N[x][y] == 3:
                Z[x][y] = 1
    return Z

show(Z)
for i in range(4):
    iterate(Z)
show(Z)
```

Exercices

1. Import the numpy package under the name `np`
2. Print the numpy version and the configuration.
3. Create a null vector of size 10
4. Create a null vector of size 10 but the fifth value which is 1
5. Create a vector with values ranging from 10 to 99
6. Create a 3x3 matrix with values ranging from 0 to 8
7. Find indices of non-zero elements from [1,2,0,0,4,0]
8. Declare a 3x3 identity matrix
9. Declare a 5x5 matrix with values 1,2,3,4 just below the diagonal

Resources

Numpy User Guide (SciPy community, 2016)

> This guide is intended as an introductory overview of NumPy and explains how to install and make use of the most important features of NumPy. For detailed reference documentation of the functions and classes contained in the package, see the NumPy Reference.

SciPy Lecture Notes (Varoquaux et al. (2015))

> The SciPy Lecture notes offers a teaching material on the scientific Python ecosystem as well as quick introduction to central tools and techniques. The different chapters each correspond to a 1 to 2 hours course with increasing level of expertise, from beginner to expert.

An introduction to Numpy and Scipy (M. Scott Shell, 2014)

> NumPy and SciPy are open-source add-on modules to Python that provide common mathematical and numerical routines in pre-compiled, fast functions. These are growing into highly mature packages that provide functionality that meets, or perhaps exceeds, that associated with common commercial software like MatLab©.

References

Oliphant, T. E. (2006). *A guide to numpy*. Trelgol Publishing.

Varoquaux, G., Gouillart, E., Vahtras, O., Haenel, V., Rougier, N. P., Gommers, R., et al. (2015). *Scipy Lecture Notes*. Zenodo.