ITCS159 Software Lab for Basic Scientific Problem Solving
Lab 6: the use of Python Library (SciPy)

September 13, 2021

Introduction

During this lab session, you will learn a basic python library for solving scientific problems named `SciPy`. `SciPy` is a collection of mathematical algorithms and convenience functions built on top of Numpy (i.e. it requires Numpy). It contains many useful modules such as optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing and ODE solvers.

SciPy Organisation

The SciPy subpackage is organized to cover different scientific computing domains. The summarization of each package are as follows:

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.
`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 algorithms.
`special`	Special functions.
`stats`	Statistical distributions and functions.

In this lab, only three sub-packages (`scipy.cluster`, `scipy.constants`, and `scipy.cluster`) will be considered for practicing.

Before start coding: Open the *Jupyter Notebook* Application. Then, go to *Desktop* and create sub folder here named *sw_lab*. Click *New* > Python3 with name *lab06*. Now you can start the following exercise.

SciPy.Cluster

In case that, we have to deal with grouping a similar things, for instance, to classify between the ripe and unripe of bananas. This action also known as clustering which is a task of grouping a set of objects that have similar characteristics, a.k.a, feature (e.g. color, shape, size). The `scipy` library in Python provides a clustering package to help you do this. K-mean is a popular clustering techniques. K-mean will be used as an example to practice how to do basic clustering using Python.

K-mean Clustering

K-mean is an algorithm for grouping similar data into K groups, where K can be specified. This algorithm returns a set of *centroids*, one for each of the clusters. Data will be transformed in to observation vector and will be classified into the closest *centroid*. Basic function of K-mean in `scipy` package are as follows:

- `whiten(obs[, check_finite])`
  Normalize a group of observations on a per feature basis.
- `vq(obs, code_book[,check_finite])`
  Assign codes from a code book to observations.
- `kmeans(obs, k_o_guess[, iter, thresh, ...])`
  Performs k-means on a set of observation vectors forming k clusters.
- `(kmeans2(data, k[, iter, thresh, minit, ...]))`
  Classify a set of observations into k clusters using the k-means algorithm.

Exercise 1) Now, try to create the final_scores of 20 students in Jupyter Notebook as follows:

```
import scipy
from scipy.cluster.vq import kmeans,vq,whiten
 #Score of 20 students in on subjects
student_scores =  [[80], [78], [55], [60], [62],
                   [59], [48], [49], [51], [54],
                   [72], [95], [78], [54], [50],
                   [61], [57], [56], [65], [53]];
```

Exercise 2) Next, normalize the raw data and calculate the centroids for classifying two group of students:

```
student_scores = whiten(student_scores)
centroids,_ = kmeans(student_scores,2)
```

Exercise 3) Finally, assign student into each group based on the nearest distance:

```
result,_ = vq(student_scores,centroids)
```

What is the result? What does it mean?

Exercise 4) Modify the code in Exercise 1,2 and 3, in order to classify students in to 4 groups and show the results.

Exercise 5) Next, Try to classify the following scores of two subjects from 20 students into FOUR groups.

```
 student_scores =
 [[80, 72], [78, 56], [55, 64], [60, 61], [62, 45],
  [59, 71], [48, 85], [49, 45], [51, 55], [54, 62],
  [72, 81], [95, 81], [78, 92], [54, 80], [50, 50],
  [61, 65], [57, 62], [56, 55], [65, 63], [53, 72]];
```

Explain the results.

Milestone 1) You have now completed the Milestone 1. You should get sign off by your lab assistant.

SciPy.Constants

The `scipy` package contains a lot of useful constants and units of both Physic and Mathematic which will allow you to easily compute scientific problems.

Exercise 6) Try the following code:

```
import scipy.constants

print("The PI is %.16f"%scipy.constants.pi)
print("The speed of light is c = %.1F"%scipy.constants.c)
print("The newton's gravity constant is G = %.1F"%scipy.
    constants.g)
```

What are the results?

Exercise 7) Try the following code:

```
scipy.constants.find("gram"))
```

What are the outputs? What is method `.find` do?

Exercise 8) Try the following code:

```
print(scipy.constants.physical_constants["atomic mass unit-
    kilogram relationship"])
```

What is the output?

Exercise 9) Try to calculate the Energy(E) of your body (weight in kg) if your entire body disappeared (Using $E = mc^2$ equation).

Milestone 2) You have now completed the Milestone 2. You should get sign off by your lab assistant.

SciPy.Stats

One of the most useful package in `scipy` library is `scipy.stats`. This package contains a large number of statistical functions. In this exercise, you will learn about the basic descriptive statistics such as Min, Max, Mean and Variance etc. Teh basic statistical functions are listed below:

Basic Statistics

- `describe()` Computes several descriptive statistics of the passed array
- `gmean()` Computes geometric mean along the specified axis.
- `hmean()` Calculates the harmonic mean along the specified axis.
- `kurtosis()` Computes the kurtosis.
- `mode()` Returns the modal value.
- `skew()` Tests the skewness of the data.
- `f_oneway()` Performs a 1-way ANOVA.
- `iqr()` Computes the interquartile range of the data along the specified axis.
- `zscore()` Calculates the z score of each value in the sample, relative to the sample mean and standard deviation.
- `sem()` Calculates the standard error of the mean (or standard error of measurement) of the values in the input array.

Exercise 10) Try the following command to find the min, max, mean and variance of the data set (19, 18, 21, 16, 15, 17, 20, 18):

```
import scipy.stats
import numpy as np
x = np.array([19,18,21,16,15,17,20,18]);
print(stats.tmin(x), stats.tmax(x), stats.tmean(x), stats.
    tvar(x));
```

Exercise 11) Let's add the following statement into the exercise 10 solution:

```
print(stats.describe(x));
```

What are the results of `stats.describe()` function?

Z-score

To measure on how many standard deviations below or above the population mean of a raw score *Z-score* is used. It also know as a standard score and can be placed on a normal distribution curve. The *Z-score* are as follow:

$z_i = \frac{x_i - \bar{x}}{S}$,Where x is a test score, $\bar{x}$ is the sample mean and S is the sample standard deviation.

Exercise 12) Using student_score of subject A as shown in Exercise 1, Try the following code:

```
import scipy.stats
import numpy as np

student_scores =  [[80], [78], [55], [60], [62],
                   [59], [48], [49], [51], [54],
                   [72], [95], [78], [54], [50],
                   [61], [57], [56], [65], [53]];

print(stats.tmean(student_scores))
print(stats.tstd(student_scores))
print((62-stats.tmean(student_scores))/stats.tstd(
    student_scores))
```

What are the results? What does it mean?

Exercise 13) Calculate Z-score of all scores in Exercise 12 (Manually). Try to use package in `scipy.stats` to calculate. How is the result?

Milestone 3) You have now completed the Milestone 3. You should get sign off by your lab assistant.