Documentation for Čech Complex Construction

This serves as (limited) documentation for the Python implementation of the Čech complex algorithm given in (Datchev and Ivrissimtzis 2012)

There are two main ways of using the algorithms:

- Directly importing the Python libraries into your project by copying the topology folder into your project

OR

- Using the supplied web services

Requirements

Requirements for using library

If you will only be using the cech code in your existing project i.e. calling the code directly, and not via webservices then you will just need to install the below software. It is all free and available for a range of operating systems.

Note that for Windows it may be easier to install a Python package that contains all of the below software. A list of these can be found on the Scipy website.

- Python 3
  - programming language the algorithms have been written in
  - https://www.python.org/downloads/
- Numpy
  - math library for Python
  - http://www.scipy.org/scipylib/download.html
- Scipy
  - contains various useful algorithms to be used with Numpy
  - http://www.scipy.org/scipylib/download.html
- pytest
  - Unit testing framework
  - http://pytest.org/latest/index.html

Requirements for using web services

If you want to be able to run the webservices so that anyone can access them via the internet then you must first find a webserver to host the services.

You can just install the required software locally on your own machine, however this will limit you to only being able to use the webservices on your machine. This is useful for testing though.

Make sure you have installed all of the requirements above for using it as a library.

Then you will need to install the following:

- RabbitMQ
  - used for long-running processes that are placed on queues
  - http://www.rabbitmq.com/download.html
- Celery
  - used to integrate with RabbitMQ via Python. Allows us to create tasks and load them into a background process
  - http://www.celeryproject.org/install/
- Flask
  - used to build webservices in Python
  - http://flask.pocoo.org
- Pandas
  - used for quickly loading csv and off files into numpy arrays
  - http://pandas.pydata.org/getpandas.html

Project Structure

The project of the structure is given in brief, most of the code is contained in similar files, so for example all of the file IO operations are kept in file.py.

- Models
  i. contains the '.off' files to calculate the cech complex from
- tests
  i. contains the unit tests for testing the algorithms developed
  - data
    i. contains test data files that the unit tests use
  - test_Cech.py
    i. tests for the overall Čech algorithm
  - test_Cells.py
    i. tests for the isMax, enumerate and recursive_enumerate methods
  - test_Entropy.py
    i. tests for the entropy and Simplical Compex Entropy methods
  - test_Sphere.py
    i. tests for the through and min_sphere methods
- topology
  i. folder containing the sub-packages and code
  - bounding
    * sphere.py
      i. contains sphere through and min_sphere methods
  - cech
    * cech.py
      i. contains the calculate method that performs the Čech method
    * cells.py

        i. contains the isMax, enumerate and recursive_enumerate method
- entropy
    * entropy.py
        i. contains a method to calculate the simplical_complex_entropy
- io
    * file.py
        i. contains methods to read a '.off' file into a set of vertices
- services
    * server.py
        i. contains the web services so that the cech, entropy and recursive_enumerate methods can be called from a separate client application
    * tasks.py
        i. sets up the methods for Celery so that they can be used in the background
- util
    * math.py
        i. file contains some math constants

Running the Unit Tests

There are a number of unit tests inlcuded in the project that test the Cech, entropy, enumeration and sphere packages.

First make sure that you have installed Pytest. Once you have installed Pytest, make sure you're in the 'Project' directory. When you are in the directory just open a console or command-prompt and enter:

```
py.test
```

This might take a little while, mainly because of the simplical_complex_entropy method that uses minimize and optimize functions that seem to be painfully slow in Python.

Using the library project

Below is an example of how to use the Python project to run the cech code if you just want to test it without copying it into your own project.

Note that the cech method has not been completely tested, so there may be errors. They should be easy enough to fix as it is very close to the original Matlab code.

It assumes that you've already installed the requirements above.

Open a console application and enter:

```
python
```

This will start the interactive Python session. Then enter the following:

```
import topology.cech.cech as cech
import topology.io.file.as off_file
import numpy as np
```

To read in the vertices from an '.off' file type in:

```
verts = off_file.read_off("Models/eight.off")
```

To run the cech algorithm enter:

```
centre_list, radius_list, basis_list = cech(verts, 0.01)
```

where 0.01 is the epsilon to be tested.

To save the output of the algorithm enter:

```
np.savetxt('centre_list.csv', centre_list)
np.savetxt('radius_list.csv', radius_list)
np.savetxt('basis_list.csv', basis_list)
```

All of the other methods are run in a similar fashion. Each of the methods in the code have been commented with the input and output parameters.

Using the web services

Starting the web services

To run the web services on your local machine, first make sure that you have all of the necessary requirements installed. Then in a console enter the following:

```
rabbitmq-server
```

This will start the rabbitmq-server, which will allow programs to be processed in the background which is useful especially when the Čech algorithm takes sometime to compute.

Ensure that you are in the directory of the Project and run:

```
python topology/services/server.py
```

This will start the webservices

Calling the Cech method

Ensure that you have the command-line app 'curl' installed, we will be using this to call the web services from the command line.

In the command-line enter:

```
curl -i -X POST http://127.0.0.1:5000/cech -F 'epsilon=0.01' -F 'vertices=@off_file_path'
```

where 'off_file_path' is the path to the off file you want to submit and 0.01 is the epsilon you wish to test

You should receive a response that looks like below:

```
HTTP/1.0 202 ACCEPTED
Content-Type: application/json
Content-Length: 2
Location: http://127.0.0.1:5000/cech_satus/4c13c05f-90cb-4192-8920-6eed4fa23850
Server: Werkzeug/0.11.9 Python/3.5.1
Date: Mon, 23 May 2016 14:22:54 GMT
```

Note that the response code is 202 ACCEPTED. This means that the cech computation has been accepted and is being processed in the background. You will not get the results immediately, you will have to call a separate method to ask if the method has finished and the results are ready to collect

Getting the results back from the Cech method

To get the results of the algorithm you will need the URL returned in the Location field of the response.

In the command-line enter:

```
curl -i -X GET http://127.0.0.1:5000/cech_satus/4c13c05f-90cb-4192-8920-6eed4fa23850
```

If the Cech algorithm is still processing then you will receive the following response:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 24
Server: Werkzeug/0.11.9 Python/3.5.1
Date: Mon, 23 May 2016 15:08:12 GMT

{
  "state": "Pending"
}
```

This indicates that the algorithm is still being processed. You will have to re-enter the request at a later time to see if the response is ready.

Eventually the program will finish, and instead of 'Pending' you will receive a response that looks like this:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 104858
Server: Werkzeug/0.11.9 Python/3.5.1
Date: Mon, 23 May 2016 16:11:51 GMT
```

```
{
  "results": {
    "basis_list": [
      [
        538.0,
        0.0,
        0.0
      ],
      [
        514.0,
        0.0,
        0.0
      ]
  ],
  ...
  }
 }
```


The output will be large, so it's easier to save it to a file which can be done by entering the below into command line, ensuring that the URL is the one returned from the location header when making the POST request.

```
curl -X GET http://127.0.0.1:5000/cech_satus/4aceb4a5-f840-4808-9661-934afb80ece1
-o file_path.json
```

where 'file_path.json' is the path to where you would like the file to be saved

Calling the Entropy method

Ensure that you have the command-line app 'curl' installed and have started the web services. In the command-line enter:

```
curl -i -X POST http://127.0.0.1:5000/entropy -F 'a=@a.csv' -F 'p=@p.csv'
```

where 'a.csv' is a comma separated file containing the A matrix and 'p.csv' is a comma separated file containg the probability values.

You should receive a response that looks like below

```
HTTP/1.0 202 ACCEPTED
Content-Type: application/json
Content-Length: 2
Location: http://127.0.0.1:5000/entropy_satus/4c13c05f-90cb-4192-8920-6eed4fa23850
Server: Werkzeug/0.11.9 Python/3.5.1
Date: Mon, 23 May 2016 14:22:54 GMT
```

This means that the request has been accepted and is being processed.

Getting the results back from the Entropy method

To get the results back from the entropy calculation you will need to use the URL returned in the Location field above.

In the command line enter:

```
http://127.0.0.1:5000/entropy_satus/4c13c05f-90cb-4192-8920-6eed4fa23850
```

where the URL is the one returned from the Location field in the POST request.

You will receive a response of the form:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 24
Server: Werkzeug/0.11.9 Python/3.5.1
Date: Mon, 23 May 2016 15:08:12 GMT

{
  "state": "Pending"
}
```

This means the request is still being processed at this time.