

Graph Theory Homework 2

Islem Rekik

Istanbul Technical University, Turkey

Submission instructions

- Copy and number the questions into your Jupyter Python Notebook. For submission, you only need to submit a zipped file named “StudentName_ID.zip” containing: (1) your Jupyter Python Notebook file: ‘StudentName_ID.ipynb’, which includes all comments and codes of the homework exercises below, and (2) the graph data to load when running the Jupyter Python Notebook.
- For those who are not used to Python notebooks, I have attached below a step-by-step suggestions on how to set up Jupyter Notebooks prepared by Dr Damien J. Duff (see last 2 pages).

1 Bellman-Ford shortest path algorithm[20%]

The Bellman-Ford algorithm is a generalization of Dijkstra’s algorithm to directed graphs with no negative cycles. Given a weighted matrix \mathbf{A} of a directed graph \mathbf{G} , an element $\mathbf{A}(i, j)$ denotes the weight of the edge *from* node i *to* node j .

- (1.1) Explain the key steps of Bellman-Ford algorithm (one-by-one). [5%]
- (1.2) Write a function called *BellmanFordAlgo* that takes (i) \mathbf{A} and (2) a starting node ID as inputs, and outputs an array as in Graph Theory Blink 5.4^a, where each row represents a node in the graph \mathbf{G} and comprising three columns (nodes, shortest distance from source input node, previous node). [10%]
Use of external libraries is not allowed. Code it up from scratch.
- (1.3) Run your function in the Jupyter Notebook on the input adjacency matrix \mathbf{A} (see **Fig. 1**) and display the output array. You can find the data inside ‘Exercise_1_data’ folder. [5%]

2 Global efficiency, diffusion efficiency and graph morphospace [40%]

This part is related to lectures 5 and 6, in particular Graph Theory Blink 6.6:

<https://www.youtube.com/watch?v=FX3Dp4ZlgzQ&list=PLug43ldmRS03MV-Jgjr30E5SpwNKkjTvJ&index=28>

Check the notes below the videos for more helpful information including:

http://basira-lab.com/wp-content/uploads/2019/11/GT_lectures_5_6_2019.pdf

^a <https://www.youtube.com/watch?v=FwxKQC-iYUg&list=PLug43ldmRS03MV-Jgjr30E5SpwNKkjTvJ&index=21>

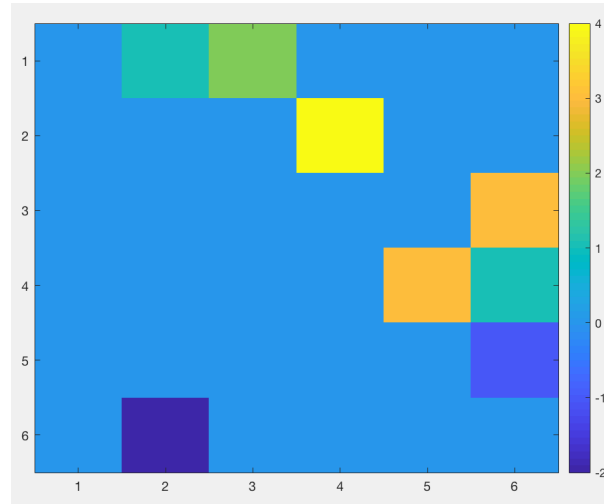


Fig. 1: The weighted adjacency matrix \mathbf{A} of the directed graph \mathbf{G} .

- (2.1) Write a function *globalEfficiency*, which inputs a graph adjacency matrix \mathbf{A} and outputs the global efficiency value. [5%]
- (2.2) Write a function *diffusionEfficiency*, which inputs a graph adjacency matrix \mathbf{A} and outputs the diffusion efficiency value. [5%]
- (2.3) Create a scatter plot where x-axis represents threshold value $\alpha \in [0 : 0.1 : 0.9]$ (0.1 denotes the threshold step size) and the y-axis represents the global efficiency E_{glob} of brain graph adjacency matrices ‘Exercise_2.data/ brainGraph1.mat’ and ‘Exercise_2.data/ brainGraph2.mat’ (Fig. 2). By thresholding each adjacency matrix at different threshold values $\alpha \in [0 : 0.1 : 0.9]$, examine how its global and diffusion efficiencies change. Use two different colors to compare the change in global efficiency across both brain graphs. [5%]
- (2.4) In a second figure, plot the diffusion efficiency E_{diff} of both matrices against α . [5%]
- (2.5) What conclusions can you derive from previous plots? Compare diffusion and global efficiencies within a single graph and across both graphs. [5%]
- (2.6) Plot a morphospace^b [1] for the set of thresholded brain graphs at $\alpha \in [0 : 0.1 : 0.9]$, where the x-axis denotes E_{diff} and y-axis denotes E_{glob} . Use two different colors to compare the two thresholded graph sets derived from each brain graph, respectively. [10%]
- (2.7) Discuss the information flow efficiency for both brain graphs based on your morphospace plot. [5%]

^b <https://www.youtube.com/watch?v=FX3Dp4ZlgzQ&list=PLug43ldmRS03MV-Jgjr30E5SpwNKkjTvJ&index=28>

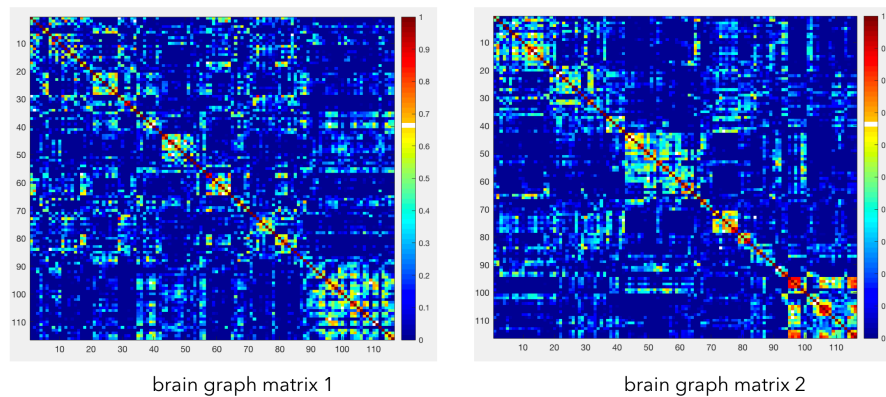


Fig. 2: *The weighted functional connectivity matrices of two brain graphs.*

3 Graph self-diffusion for image segmentation [2] [40%]

This part is related to lecture 7: <https://www.youtube.com/watch?v=h-ru40T6SGU&list=PLug43ldmRS03MV-Jgjr30E5SpwNKKjTvJ&index=29> and the research paper on ‘Affinity learning via self-diffusion for image segmentation and clustering’ by Wang et al. [2].

- (3.1) Write a function called *selfDiffuse* that takes (i) a weighted graph adjacency matrix (i.e., similarity matrix) \mathbf{W} and outputs the diffused matrix \mathbf{W}^* (check algorithm 3.). Make sure that your algorithm automatically sets the optimal number of diffusion iterations t^* as explained in the paper [2]. [10%]
- (3.2) Run *selfDiffuse* on both brain graph adjacency matrices ‘Exercise.2_data/ brainGraph1.mat’ and ‘Exercise.2_data/ brainGraph2.mat’. For each brain graph, visualize both original and diffused matrices. [10%]
- (3.3) Given the 2 images taken from the Berkeley Segmentation Data Set^c, generate the segmentation maps of each image using the Normalized Cut Python code <https://github.com/marktao99/python/blob/master/CVP/samples/ncut.py>. Display each original image and its corresponding output segmentation map. [10%]
- (3.4) For each image, change the Normalized Cut Python by applying *selfDiffuse* to the similarity matrix \mathbf{W} , then use \mathbf{W}^* to perform the normalized cut and output the image segmentation map. For each image, display 5 different normalized cut segmentation maps when varying the diffusion threshold from $t = 1$ to $t = 2 \times t^*$ (similar to Fig 1 in [2] and Fig 4 below). [10%]

^c <https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html>

1. Computing the smoothing kernel:

$$P = D^{-1}W$$

where D is a diagonal matrix with $D(i, i) = \sum_{k=1}^n W(i, k)$.

2. Performing smoothing for t^* steps:

$$W_t = W_{t-1}P + I.$$

3. Self-normalization: $W^* = W_t * D^{-1}$

Figure 2: Algorithm of self-diffusion (SD).

Fig. 3: *Graph self-diffusion algorithm to implement [2].*

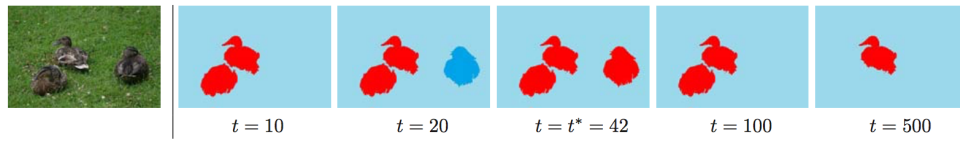


Fig. 4: *A qualitative comparison of segmentation results via self-diffusion w.r.t. different number of iterations [2].*

References

1. Goni, J., Avena-Koenigsberger, A., de Mendizabal, N.V., van den Heuvel, M.P., Betzel, R.F., Sporns, O.: Exploring the morphospace of communication efficiency in complex networks. *PLoS One* **8** (2013) e58070
2. Wang, B., Tu, Z.: Affinity learning via self-diffusion for image segmentation and clustering. 2012 IEEE Conference on Computer Vision and Pattern Recognition (2012) 2312–2319

Submission Procedure

- Only electronic submissions through *Ninova* will be accepted.
- Late submissions or those submitted otherwise than according to instructions will not be accepted.
- Submit:
 - A **ipynb** file containing the final program (Python 3 or Cling/C++17) and its documentation.
- Submit in English.
- Academic dishonesty, including cheating, plagiarism, and direct copying, is unacceptable.

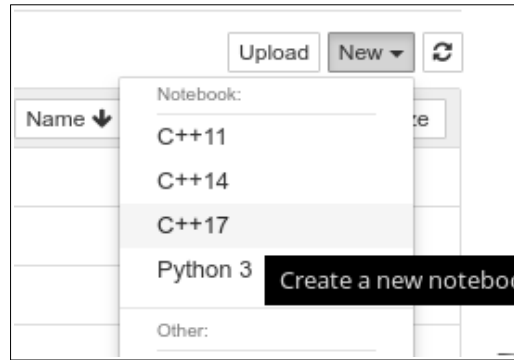
Step-by-step suggestions

If you are confident enough you may skip the below step-by-step suggestions and do your own thing but if you do that, do read these instructions anyway to ensure you have not missed something.

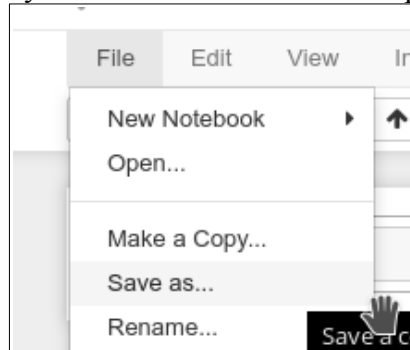
Setup

The following setup instructions might be helpful.

1. It is possible to use Jupyter Notebooks entirely in the [cloud](#) (e.g. AWS, Google Collab, Azure, Binder, CoCalc, Paperspace... none of which has been tested for this assignment). However, you may wish to install them on your own computer (this has been tested on (K/L/X)Ubuntu variants 16.04 and 18.04). Installing Jupyter Notebooks is best done with [Miniconda](#) (note that a full Anaconda install is not recommended because of some incompatibilities with the experimental C++ interpreter) on all platforms (including Linux and Windows – support may only be available for Linux). The below commands install Jupyter Notebooks on Ubuntu variants 16.04 and 18.04 with both Python 3 and C++17 kernels, assuming that Miniconda is already installed and in the system path:
`conda create -n blg443 # create a conda environment for our course`
`source activate blg443 # activate that`
`# The next two lines install the C++ interpreter for Jupyter`
`conda install xeus-cling xplot -c QuantStack -c conda-forge`
`conda install widgetsnbextension bqplot -c conda-forge`
`# The next line installs Jupyter with comes with a Python 3 kernel`
`conda install jupyter matplotlib`
2. Once you have installed Jupyter you can start a notebook (at least on Linux) using the command:
`jupyter notebook`
You can start a notebook in the language of your choice by clicking on New and selecting the appropriate language (either Python 3 or C++17 for this course):



3. You can save your notebook at any time for submission using the File.. Save As.. functionality (if you are working in the cloud you may need to use the Download option):



4. For more information and example Jupyter notebooks for **Python** see the following examples (see the next step for C++):

New to Python?

- <https://www.learnpython.org/>

Want to see the basics of working with Jupyter and Python? See:

- <https://nbviewer.jupyter.org/github/jupyter/notebook/blob/master/docs/source/examples/Notebook/Notebook%20Basics.ipynb>
- <http://opentechschoool.github.io/python-data-intro/core/notebook.html>

Want to see what Jupyter+Python can offer in diagrams?

- <https://nbviewer.jupyter.org/github/ipython/ipython/blob/1.x/examples/notebooks/Part%203%20-%20Plotting%20with%20Matplotlib.ipynb>
- <https://nbviewer.jupyter.org/github/jrjohansson/scientific-python-lectures/blob/master/Lecture-4-Matplotlib.ipynb>

Adding documentation to Jupyter notebooks:

- <https://nbviewer.jupyter.org/github/ipython/ipython/blob/1.x/examples/notebooks/Part%204%20-%20Markdown%20Cells.ipynb>

5. For more information and example Jupyter notebooks for **C++** see the following examples (see the previous step for Python):

Want to see the basics of working with Jupyter and C++? See:

- <https://nbviewer.jupyter.org/github/jupyter/notebook/blob/master/docs/source/examples/Notebook/Notebook%20Basics.ipynb>
- <https://mybinder.org/v2/gh/QuantStack/xeus-cling/stable?filepath=notebooks/xcpp.ipynb>

Want to see what Jupyter and C++ can offer in diagrams? See in particular the examples here:

- <https://mybinder.org/v2/gh/QuantStack/xplot/stable?filepath=notebooks>

Adding documentation to Jupyter notebooks: