BU.330.760 Deep Learning with Unstructured Data

Lab 3. graph-tool and Stochastic Block Models

MacOS Version

Learning Goal: practice using graph-tool package to infer modular network structures
Required Skills: knowledge on stochastic block models and hierarchical stochastic block models

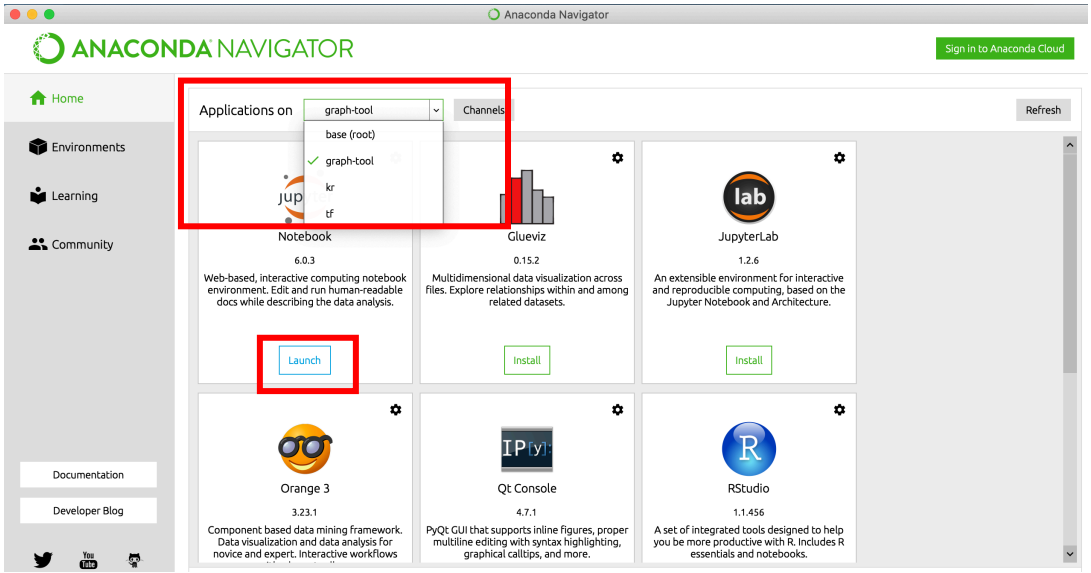
We will use graph-tool (https://graph-tool.skewed.de) in lab 3, which is an efficient Python module for manipulation and statistical analysis of graphs/networks. **But graph-tool was tested extensively *only* on GNU/Linux and MacOS X systems,** so you need to follow different instructions based on your operating system. This document is for MacOS system users, or those having MacOS virtual machine set up on Windows system.

1. Install graph-tool package with **Aanconda Navigator > JupyterLab > Launcher > Terminal**. Type in:

```
conda create -n graph-tool
conda activate graph-tool
conda install -c conda-forge graph-tool
```

This will create a new environment for graph-tool and then install it. On Anaconda, you can have multiple environments having different versions of Python and/or packages installed in them.

2. After installation, switch to the graph-tool environment to use the package (as shown in the screenshot below). You may need to install other tools in this environment as well. I recommend installing Jupyter notebook, which will be enough for this course. You will see "Launch" instead of "Install" when Jupyter notebook is ready to use.



3. If you prefer to use your own Python developer tools or platforms, please install graph-tool package according to instruction: https://git.skewed.de/count0/graph-tool/-/wikis/installation-instructions.
4. In Jupyter notebook, or your own Python development environment, build the following scripts

    a. Import required packages.

    ```
    import graph_tool.all as gt
    import matplotlib.pyplot as plt
    ```

    b. We will use a network of American football teams loaded from graph-tool's dataset collection module(https://graph-tool.skewed.de/static/doc/collection.html).

    ```
    g = gt.collection.data["football"]
    print(g)
    ```

    c. Visualize this network.

    ```
    gt.graph_draw(g, vertex_text=g.vp.label, vertex_font_size=10,
                  output_size=(1200, 1200))
    ```

    d. Apply stochastic block models on this network.

    ```
    state = gt.minimize_blockmodel_dl(g)
    print(state)
    ```

    Here SBM returns a **BlockState** object that includes the inference results. And this **BlockState** object is stored in an instance called "state". Note that SBM (an inference algorithm) used is stochastic by nature, and may return a different answer each time it is run.

    e. Draw the partitions according to SBM result.

    ```
    state.draw(pos=g.vp.pos)
    ```

    you may also use *output* parameter to save the drawing to a .png file. In a markdown cell, answer:

    Q1: Compared the block model plot to the original network plot in step c, how are they different? What's the information you can get from the block model plot?

    f. Now let's draw the heat map, we will also display the edge counts with-in, and cross blocks.

    ```
    e = state.get_matrix().todense()
    fig, ax = plt.subplots()
    im=ax.imshow(e)
    n=state.get_B()
    for i in range(n):
        for j in range(n):
            text = ax.text(j, i, int(e[i, j]),
    ```

```
                        ha="center", va="center", color="w", size="8")
fig.tight_layout()
```

You can also use **matplotlib**'s *savefig()* function to save the plot to an .png file. In a markdown cell, answer:

Q2: What information can you get from this heat map? Are there any assortative blocks? Any disassortative blocks? Is the result consistent with the block plot in step e?

5. Assignment questions: analyze a more complicated network using hierarchical stochastic block models. The network you will use is "polblogs" which is also from graph-tool dataset collection module. This is a directed network of hyperlinks between weblogs on US politics, recorded in 2005. Since this is a larger network, you do not need to visualize it. Do the following steps:

    a. Apply hierarchical stochastic block model.

Hint 1: use graph-tool function *minimize_nested_blockmodel_dl* instead of *minimize_blockmodel_dl* for hierarchical SBM. This will return an instance of a **NestedBlockState** class, which encapsulates the results. The hierarchical SBM (especially if carried out on complex networks) may take time to complete the modeling process, so you need to be patient.

    b. Draw the hierarchical block plot.

Hint 2: *draw()* function with no parameters can serve the purpose.

    c. Print the summary of hierarchical SBM, as well as summaries at each level.

Hint 3: **NestedBlockState** object has a function *print_summary()*.
Hint 4: To fetch leveled results, you need to use **NestedBlockState**'s function *get_levels()*. And then, you can use a *for* loop to print each level, such as:

```
levels = state.get_levels()
for s in levels:
    print(s)
```

    d. Finally, draw the heat maps at level 1 and 2.

Hint 5: you need to index into levels and then create the heat map plots. If you use instance "levels" to store the levels from the **NestedBlockState** instance, these will be *levels[1]* and *levels[2]*, and each is a **BlockState** instance.

In a markdown cell, answer:

Q1:How many levels are there in your model? How many blocks are there in each level?

Q2:From the heat maps of two levels, which blocks are assortative? Which are disassortative?

Submission:

Finish all the steps in the lab, save it to lab3.ipynb file with all the outputs, as well as your answers to Q1-Q2 in markdown cells. Label each markdown cell with question number, such as *Q1: your answer*.

Complete the assignment questions, save it to assignment3.ipynb file with all the outputs, and your answers to Q1-Q2 in markdown cells. Do not forget to label your answers.

Your submission should include

1. lab3.ipynb and assignment3.ipynb, if you use Jupyter notebook, or
2. lab3.py and assignment3.py, together with all outputs and a plain text file with all the answers, if you use other Python development tools. Note that all outputs and answers should be labeled.

Due: Feb 15th 11:30 am EST

Reference:

graph-tool 2.31 Documentation