COMP90051 Statistical Machine Learning

Semester 2, 2015

Workshop Week 11: Community Detection

Introduction

In this workshop you will get an experience with detecting communities in graphs using two approaches, edge betweenness and clique percolation. We will be using a MATLAB implementation of these methods, and therefore, you will need to complete the steps of the workshop in MATLAB. If you have never worked with MATLAB before, don't worry: we will only need very basic steps; there are plenty of quick start tutorials out there; and you are always welcome to ask for help from the tutor. It is always fun to learn a new programming language!

The graph that we will be working with is the Karate club network mentioned in the lectures. You should open the lecture notes so that you have a visual representation of the graph. Recall, that in this particular graph, the real community structure is known. This is because eventually the Karate club was split, and the division of members can be considered indicative of community structure with two communities. This is what is shown in circles and squares in the lecture notes.

Following the steps of this workshop might be enough even if you have no prior MATLAB experience. If you do prefer to go through a MATLAB tutorial first, you can start with the following link:

http://web.eecs.umich.edu/~aey/eecs451/matlab.pdf

(Alternatively, simply search for any other tutorial on the Internet.)

Step 1

a. Download and unpack the code and data package from LMS weekly schedule page.
b. Open MATLAB. MATLAB is an integrated development environment that by default will open a console, workspace and list of files and folders in the current path. In addition, at the top of the MATLAB main window there will be a bar with the current path. Use this bar (or a "browse folder" icon next to it) to navigate to the folder with extracted code and data.
c. Load the graph into workspace. To do this, simply drag and drop `karate.mat` into the workspace or console. You will notice a variable `mKarate` appeared in the workspace. This variable contains an adjacency matrix for the Karate club graph. You will be able to pass this variable as an input to the community detection algorithms.

Step 2

a. Navigate to subfolder called `betweenness`, either by using the current folder bar as before, or by double clicking this subfolder in the Current Folder window.
b. Double click `newmangirvan.m` file to open it in the editor. Read the description at the top of the file. Note that lines starting with '%' are comments.
c. In order to use the edge betweenness method to find 2 non-overlapping communities, type the following command in the Command Window (console)

```
>> modules = newmangirvan(mKarate, 2);
```

d. Note that the function returns three variables, but we will only focus on the first one in this workshop. In MATLAB, subsequent variables can be omitted if they are not used. Also there is no need to declare variables in advance.
e. You can now explore the resulting communities by double clicking variable `modules` in Workspace. You will find that this variable consists of two further arrays (one per community) and double clicking each of them will bring up a list of vertices that belong to each of the communities. Compare these memberships to the reference community structure shown in the lecture notes (by circles and squares).
f. Run the algorithm several times, altering the number of requested communities from 2 to 6. Note the observed effect and try to explain it (think about the dataset and how algorithm operates).
g. Do you think this is an appropriate algorithm for finding communities in this graph?

Step 3

a. Navigate to directory `k-clique`. Open `k_clique.m` and read the description of this function.
b. Run `k_clique` function to search for communities using a clique size of 2. Explore the output. How many communities were found? How do these communities align with the reference communities?
c. Run the algorithm several more times varying the requested clique size from 2 to 6. Note the observed effect and explain it (think about the dataset and the type of structure the algorithm is looking for).
d. Do you think this is an appropriate algorithm for finding communities in this graph?

SPOILER ALERT: Read this only after attempting steps 1 – 3 on your own.

Discussion

There is no solution as such for this workshop. Instead I'm providing a couple of thoughts on the observed effect and appropriateness of each of the algorithms.

Edge betweenness

```
>> modules = newmangirvan(mKarate, 2);
```

Invoking the function as above finds two communities that resemble the "ground truth" communities fairly well (see Deck 20, slide 11). So the algorithm is appropriate for this particular graph.

As was discussed in the lectures, this algorithm actually produces a hierarchy of clusters. By specifying a required number of communities when calling the function, one recovers the grouping into communities at a particular level of hierarchy (in fact, the complete hierarchical tree is shown in Deck 20, slide 11). This algorithm does not estimate the "best" number of communities, so one has to experiment with different values and also inspect the tree.

Clique percolation

```
>> modules = k_clique(5, mKarate);
```

There are no cliques larger than 5 in this graph, so invoking this function requesting the clique size of 6 or more results in no communities being found. With 5-cliques, the algorithm finds one community (a single 5-clique of vertices 1, 2, 3, 4, 14), and with 6-cliques, the algorithm returns 3 communities. Of these 3, the two are overlapping and belong to one of the true communities, and the 3rd one is from the other true community. In a way, the clique percolation method tends to identify some community cores in this graph, rather than partitioning the graph. Perhaps, this algorithm could be used to "seed" community cores and then additional "expanding" the cores to periphery is required. Therefore, clique percolation method is less suitable for this graph. A particular disadvantage is that one cannot directly require partitioning into a certain number of communities (e.g., in two). This split is indirectly controlled by the clique size.