BMI 826. Computational network biology. Homework 3
Due Dec 15th, 2018

Instructions

The goal of this homework is to get some practical experience with spectral clustering and diffusion on graphs. We will again use MATLAB code to do this homework. Please follow these instructions to get the problem 1 and 2 specific files. Please email your homework solutions to Prof. Roy.

1. Download the HW3 related files from `http://pages.discovery.wisc.edu/~sroy/teaching/network_biology/fall2018/homeworks/hw3.tgz`. You would need to run these programs on a computer that has MATLAB installed. The class servers `mi1.biostat.wisc.edu` and `mi2.biostat.wisc.edu` should have MATLAB installed.
2. Unzip this file as follows `tar -xvzf hw3.tgz`. This will result in the directory `hw3` which will have two directories, `prob1` and `prob2`.
3. Execute the MATLAB scripts listed below as described. For each script, you can type `help <script_name>` and you should be able to see a basic explanation of the inputs of the script.

Problem 1 (25 points). Effect of different types of similarity graphs on spectral clustering.

Recall that in spectral clustering we wish to cluster objects based on their connectivity on a similarity graph. In this question we will see how the results of clustering change as we change the similarity graph. This graph can be created using three ways: (a) ϵ-neighborhood graph, (b) k nearest neighbors, (c) fully connected graph. The ϵ-neighborhood graph creates a graph which connects two nodes that are within and ϵ distance of each other. The graph is unweighted. The k nearest neighbor graph adds an edge between node x and y if x is among y's k closest neighbors or vice versa. The graph is weighted with weights corresponding to the similarity of each pair of vertices based on their attributes. The fully connected graph is a weighted, completely connected graph with edge weights corresponding to their similarity.

Problem 1 instructions

1. Set current directory to `prob1`. Start MATLAB by typing `matlab`
2. List the contents of `prob1` as `ls` and you should see two directories `code` and `data`. `code` has the list of MATLAB programs that can be executed within the MATLAB prompt.

3. Execute `addpath(code)` in `matlab`. `data` has two files, `dataset1.txt`, which and `dataset1_labels.tx` which has the true cluster labels.

4. This is an optional step, but might help to have all your output files in one directory. `mkdir outputs`. Specify this as the directory for your output file paths if you do this step.

Problem description

(a) Apply the matlab script, `createSimilarityGraph.m` on `data/dataset1.txt` with different options for graph types to create different similarity graphs. This function takes four arguments: 1) the name of the input dataset file, 2) the output path where the similarity graph will be written, 3) the type of graph to be generated, 4) a parameter whose value depends on what graph type is specified in (3). The value of the third argument can be `knn`, `eps` or `fullyconnected`. If the third argument is `fullyconnected`, the fourth argument should be `[]`. Otherwise for `knn` this should be the number of neighbors and for `eps`, it should be the ϵ-neighborhood. An example usage of this function to generate the k-nearest neighbor graph is

`createSimilarityGraph('data/dataset1.txt','outputs/knn_5.txt','knn',5).`
Use different values of $k \in \{2, 4, 6\}$ to create different k-nearest neighbor graphs.

(b) Use matlab script `spectralcluster.m` to cluster the graphs created in (a). This script takes in three arguments, `spectralcluster('inputgraph.txt',k, 'output.txt'` and outputs the clusters in `'output.txt'`. The k is the number of clusters.

(c) Apply the matlab function, `compareclusters.m` to assess how similar the clusters you get to the original set of clusters in `data/dataset1_labels.txt`. This script will take in the path to the true cluster IDs and the path to the inferred cluster ID. It will output several numbers, each specifying a pairwise best similarity between one of the inferred and one of the true clusters. The `Average similarity` number is what can be used to assess the overall similarity. This is a number between 0 and 1 and the closet it is to 1, the higher the similarity. Apply this script to all your inferred cluster assignments and provide the corresponding average similarities for each k. Discuss your results in terms of the choice of k.

(d) Apply steps (a) but using the ϵ-neighborhood graph with $\epsilon \in \{0.1, 0.5\}$. Repeat step (b) on these two graphs. Repeat step (c) on the inferred cluster assignments. Please provide the corresponding average similarities and discuss these results in the context of different values ϵ.

(e) Apply steps (a) and (b) for the `fullyconnected` option of `createSimilarity` function. Discuss your result.

(f) Apply the function `viewcluster.m` to visualize the inferred and true clusters for `knn` graph for $k = 2$, `eps` graph for ϵ=0.1 and ϵ=0.5, and for the `fullyconnected` graph. Discuss your observations together with the values you get from applying the `compareclusters.m` script. Which type of graph might you choose? Justify your choice.

Problem 2 (25 points). Effect of different kernels on diffusion

We have seen different kernels to perform diffusion on graphs. These kernels can be used to prioritize genes as well as to smooth out matrices. The goal of this question is to study the different kernels and their

parameters to see the effect on diffusion results.

Problem 2 instructions These instructions are similar to Problem 1.

1. Set current directory to `prob2`. Start MATLAB by typing `matlab`
2. List the contents of `prob2` as `ls prob2` and you should see two directories `code` and `data`. `code` has the list of MATLAB programs that can be executed within the MATLAB prompt.
3. Execute `addpath(code)` in `matlab`.

Problem 2 description You are provided with the matlab script `networkDiffusion.m`, which can perform diffusion on the network using different kernels and output the diffusion score and rank on all nodes in the network. This function takes five arguments, `networkDiffusion(inputgraph, query,kernel, kernel_param,outputfile)`. `inputgraph` is the file path of the file with the input binary adjacency matrix. `query` is the file path for the file containing the query nodes. This is a text file with rows corresponding to nodes in the network and two columns. The first column has the node name and the second column is 1 for all the nodes that query nodes. `kernel` is the name of the graph kernel we want to use and we can choose from four: (a) `rwr` for Random walk kernel, (b) `diffusion` for a diffusion kernel from the Kohler et al paper, (c) `heat1` which is the first Hotnet kernel, and (d) `heat2` which is the second Hotnet kernel. The fourth argument is the kernel specific parameter. The fifth argument is the location of the output file where the node rankings will be written out.

(a) Apply the `networkDiffusion` function with the `rwr` kernel on the input graph, `data/graph1.txt` and `data/query.txt` as the query file. For random restart probability, try values of $\{0.1, 0.3, 0.5\}$ and write the results out in a separate output file. An example usage is: `networkDiffusion('data/graph1.txt','data/query.txt','rwr',0.1, '../outputs/prob2/rwr_0.1.txt');`. where the restart probability $= 0.1$ and output rankings are written to the file `../outputs/prob2/rwr_0.1.txt`.

(b) Use the matlab function `compare_pcc.m` function provided to assess how similar the rankings are to each other and please provide the values for each pair of comparison. Describe a strategy to determine what restart value you can use and mention the restart value you will use

(c) Repeat (a) with the `diffusion` kernel. Recall that the diffusion kernel is defined as $exp(-\beta L)$ where β is the kernel parameter controlling the strength of diffusion. Set β to $\{0.1, 0.3, 0.5\}$ and apply the `compare_pcc.m` function to obtain the similarity of the rankings.

(d) Repeat (a) with the two heat kernels, `heat1` and `heat2`. `heat1` is defined as $L + \gamma I$, and `heat2` is defined as $\beta(I - (1 - \beta)W)^{-1}$. Here L is the graph Laplacian and W is the adjacency matrix. Experiment with different values of $\gamma \in \{2, 5\}$ and different values of $\beta \in \{0.1, 0.3\}$. Use the `compare_pcc.m` script to report the similarity of the rankings from the two kernels across the different parameter settings.

(e) Examine the inferred ranks of the query nodes in the output files. A kernel which gives high ranks to the input nodes is likely the best. Based on this, mention which kernel you might choose.

(f) Compare the different output rankings across the four kernels and their input parameter settings using the `compare_pcc.m` script. Specifically, provide a table of correlation values comparing all pairs of rankings from the 10 rankings generated using the different kernel and parameter configurations. Based on the correlation values, which kernels appear most similar to each other?