Discovery Thread: Project 2

In this project you will apply the techniques for random graphs model selection and community detection on a specific data set.

Three files are assigned to your teams: a DataFile *.dat , and two images *.bmp as described next.

1. The image file Phantom*.bmp contains a $n \times n$ black-and-white image that is utilized as reference signal for your project. You can load this image in Matlab using imread() function. Note the entries are unsigned 8-bit integers (hence in the range 0 .. 255).

2. The graph weight matrix file WeightMatrixForImage*Noisy.dat . This text file has the following format:

   ```
   First line: n
   Second line: W(1,1)  W(1,2)  W(1,3)  ...  W(1,n)
   Third line:  W(2,1)  W(2,2)  W(2,3)  ...  W(2,n)
   ...
   Line n+1: W(n,1)  W(n,2)  W(n,3)  ...  W(n,n)
   ```

   where the entries $W(i, j)$ were computed based on the reference Phantom image. Specifically $W(i, j) = exp(-|I(i) - I(j)|/20 - \|i - j\|^2/10)$ where $I(i), I(j)$ are noisy intensities of pixels i and j of image Phantom*.bmp.

3. The image HighResNoisyPhantom*.bmp is the upsampled noisy reference image Phantom*.bmp.

In your project $n = 1024$ and corresponds to a 32×32 image. Vertices are pixels indexed row-wise from left to right:

$$vertex_1 = pixel(1, 1)\ ,\ vertex_2 = pixel(1, 2)\ ,\ vertex_3 = pixel(1, 3)\ \cdots\ vertex_{33} = pixel(2, 1)\ \cdots$$

On the weighted undirected graph dataset assigned to your project perform the following three tasks:

I. Random graph model testing:

Point Estimation:

1. Under the Erdos-Renyi random graph model, estimate the parameter p. Compute the estimated number of 3-cliques and 4-cliques and compare them to the actual numbers of 3-cliques and 4-cliques in your data set.

2. Under the SSBM random graph model, estimate the parameters a and b based on the number of vertices, edges, and 3-cliques. Compute the estimated number of 4-cliques (under the SSBM model) and compare this predicted number to the actual number of 4-cliques.

Sequence of 4-cliques prediction:

1. Create an ordered sequence of edges accoring to their weight. Specifically, order the edges according to the weight, starting with the largest weight first and then continue in a monotonic decreasing order. To do so, create a data file, say graph.dat, from the data file assigned to your project, that lists the edges in the appropriate order, and has the following format:

```
First line: n m
Second line: Edge1Vertex1 Edge1Vertex2
Third line: Edge2Vertex1 Edge2Vertex2
...
m+1st line: EdgemVertex1 EdgemVertex2
```

2. For the sequence of edges (and graphs) perform the following computations:

   (a) Under the Erdos-Renyi random graph model, for each graph in the sequence, estimate the parameter p, and compute the expectation of the number of 4-cliques; On the log-log plot, determine the best linear fit, $log(X_4) = a_{ER}log(m) + b_{ER}$, where m is the running number of edges; discard the first values of m when there are no 4-cliques.

   (b) Under the SSBM random graph model, for each graph in the sequence, estimate parameters a and b and compute the expectation of the number of 4-cliques; On the log-log plot, determine the best linear fit, $log(X_4) = a_{SSBM}log(m) + b_{SSBM}$, where m is the running number of edges; discard the first values of m when there are no 4-cliques.

   (c) For each graph in the sequence, compute the actual number of 4-cliques, $X_4(m)$, and determine the best linear fit, $log(X_4) = a_0log(m)+ b_0$, where m is the running number of edges; discard the first values of m when there are no 4-cliques.

3. Overlay in the same plot the graphs of $log(X_4)$ and the prediction under Erdos-Renyi and SSBM models of the number of 4-cliques. Print also the parameters a_{ER}, a_{SSBM}, a_0 and b_{ER}, b_{SSBM}, b_0.

II. Community detection:

Implement the six community discovery algorithms (partition algorithms) and run them on your project data set.

Specifically, implement:

1. Spectral methods using: W, Δ, and $\tilde{\Delta}$

2. SDP relaxation algorithms using: W, Δ, and $\tilde{\Delta}$

Recall rasterisation is done row by row.

For each of the six algorithms above, determine sets S and $\bar{S} = \{1, 2, \ldots, n\} \setminus S$. Then map back onto the noisy image the two sets, and overlay the resulting partition.

For easy visualization, use the High resolution image (which is 1024×1024 pixels). Each pixel in the standard (low-res) 32×32 image is upsampled to (i.e., replaced by) a 32×32 block of pixels in the High resolution image. For each algorithm present the following images images:

1. The High resolution noisy and/or clean image
2. The indicator image of the partition sets, as a two-tone image;
3. Overlay of the partition indicator to the noisy High resolution image