Ioana-Sabina STOIAN
sabina.stoian@gmail.com

1 Question 1

The maximum number of edges that an undirected graph of n nodes without self-loops can have is equal to:

$$C_n^2 = \frac{n(n-1)}{2} \tag{1}$$

The maximum number of triangles that an undirected graph of n nodes without self-loops can have is equal to:

$$C_n^3 = \frac{n(n-1)(n-2)}{6} \tag{2}$$

2 Question 2

The histogram of the nodes' degrees from the CA-HepTh graph is presented in Figure 1. As we can observe from this figure, the nodes' degrees are distributed according to an **exponential distribution**. The frequency of lower degrees surpasses considerably the frequency of the higher ones. More precisely, the frequency decreases exponentially as the degree of the node increases.
This is an indicator about the connectivity of nodes within the graph. In a fully connected graph, all the nodes will have the maximum degree, while in an empty graph, all the nodes will have the degree 0. Thus, every other type of graph will be in between these two extremes. By inspecting Figure 1, because the frequency when moving towards low-degree nodes is growing exponentially, this might be an indicator that the graph is weakly connected. The same remark can be made by analysing the number of edges from the graph (25.998 edges) and the number of edges that a fully connected graph, with the same number of nodes (9.877 nodes), contains. By applying Eq. 1, the fully-connected graph will have $48.772.626$ edges, which is considerably higher than the number of edges in the analysed graph.

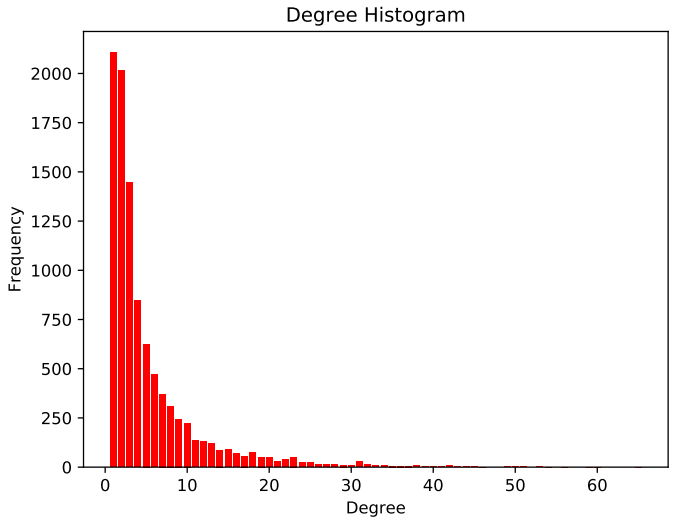

Figure 1: The histogram of the nodes' degrees from the CA-HepTh graph.

3 Question 3

The answer to this question is based mostly on [4], as well as on [3] and [2].
In order to obtain a proper partition of the data, a clustering algorithm would ideally find the solution that maximizes the intra-connectivity in the same cluster (the connectivity within the same community is high) and that minimizes the inter-connectivity with other clusters (the number of edges between nodes from different clusters should be kept low). The problem that is optimized by the eigenvalue decomposition used in the

spectral clustering algorithm can be interpreted as a relaxation of a graph cut problem formulation, which would otherwise be intractable.
[4] presents in more details the link between the spectral clustering algorithm and an optimization problem which can be interpreted as a relaxed variant of minRatioCut. Minimizing the sum of the weights of edges between clusters, which is desirable for clustering algorithms, corresponds to a graph cut problem. To create more balanced clusters, instead of the original graph cut problem, a related variant called minRatioCut can be solved. While solving minRatioCut exactly is NP-hard, the problem can be relaxed so as to be made tractable. In the relaxed formulation, minRatioCut for $k \geq 2$ corresponds to finding the k smallest eigenvectors of the Laplacian, which represent the eigenvectors returned by the spectral clustering algorithm. The discrete partitions of the graph corresponding to the k clusters can then be obtained by using K-means clustering on the rows of the U matrix, containing as columns the eigenvectors (corresponding to the smallest k eigenvalues). This is precisely the formulation of the spectral clustering algorithm.

4 Question 4

Number of edges within the graph:

$$m = |E| = 10 \tag{3}$$

Number of clusters

$$n_c = 3 \tag{4}$$

For the cluster 1:
Number pf edges within community c_1 (the green cluster):

$$l_{c1} = 1 \tag{5}$$

Sum of degrees of the nodes that belong to community c_1 (the green cluster):

$$d_{c1} = d_1 + d_2 = 1 + 1 = 2 \tag{6}$$

For the cluster 2:
Number of edges within community c_2 (the blue cluster):

$$l_{c2} = 3 \tag{7}$$

Sum of degrees of the nodes that belong to community c_2 (the blue cluster):

$$d_{c2} = d_3 + d_4 + d_5 = 2 + 2 + 3 = 7 \tag{8}$$

For the cluster 3:
Number of edges within community c_3 (the gray cluster):

$$l_{c3} = 5 \tag{9}$$

Sum of degrees of the nodes that belong to community c_3 (the gray cluster):

$$d_{c3} = d_6 + d_7 + d_8 + d_9 = 3 + 2 + 3 + 3 = 11 \tag{10}$$

The modularity of the graph Q will be computed in the following way:

$$\begin{aligned} Q &= \sum_{i=1}^{3} \left[\frac{l_{ci}}{m} - \left(\frac{d_{ci}}{2m} \right)^2 \right] \\ &= \left[\frac{1}{10} - \left(\frac{2}{20} \right)^2 \right] + \left[\frac{3}{10} - \left(\frac{7}{20} \right)^2 \right] + \left[\frac{5}{10} - \left(\frac{11}{20} \right)^2 \right] \\ &= 0.465 \end{aligned}$$

5 Question 5

The answer to this question is based on [1]. Two examples of graphs that are non-isomorphic and which are mapped to the same representation by the shortest path kernel are illustrated in Figures 2 and 3. To justify that these two graphs are not isomorphic, we will analyse how the nodes are connected in the two graphs. G_1 has 2 cycles consisting of 3 nodes, each of it having a degree of 3, whereas G_2 does not have this type of cycle (graph G_2 only has cycles consisting of a pair of nodes). Therefore, the two graphs are non-isomorphic.

In the following, we will compute the mapping that will result after applying the shortest path kernel on each of the graphs. The first graph, G_1, has 9 shortest paths of length 1 and 6 shortest paths of length 2. Therefore, the feature vector resulted after applying the shortest path kernel is $\phi(G_1) = [9, 6, 0, ..., 0]$. After repeating the procedure for the second graph, G_2, we will end up having the exact mapping ($\phi(G_2) = [9, 6, 0, ..., 0]$).
Thus, the two graphs, G_1 from Figure 2 and G_2 from Figure 3 are non-isomorphic, but they are mapped to the same feature vector by the shortest path kernel.

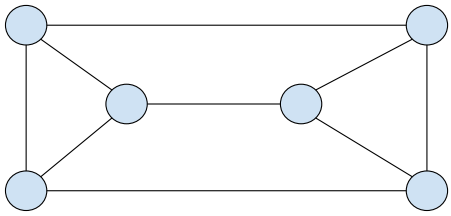

Figure 2: Graph G1: the shortest path kernel maps the graph G1 to the feature vector: $\phi(G_1) = [9, 6, 0, ..., 0]$.

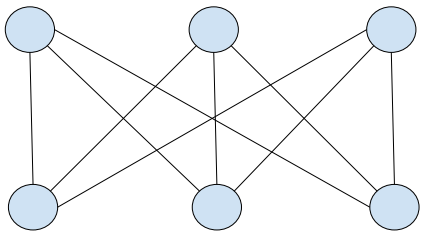

Figure 3: Graph G2: the shortest path kernel maps the graph G2 to the feature vector: $\phi(G_2) = [9, 6, 0, ..., 0]$.

6 Question 6

The performances of the two graph kernels are depicted in Table 1.
We can see from Table 1 that the shortest path kernel surpasses considerably the graphlet kernel, as with the first method we achieve a perfect accuracy score and with the latter one we obtain a score that is comparable to that of a random classifier. Intuitively, the reason that the graphlet kernel may perform poorly on this problem is that the instances with the same number of nodes of the two classes of graphs differ structurally from one another because of only two vertices, which are connected in the cycle graph and which are not connected in the path graph (the vertices of degree one from the path graph). Because at every iteration using the graphlet kernel we sample randomly from a graph groups of three nodes, it is less probable to sample the exact those nodes that differ from a class to another, especially when the number of nodes in the graph is relatively high. The most probable scenario for one sampling step is that we will sample nodes that are not connected to one another and the subgraph created in this way will be isomorphic with the empty graphlet (the G4 graphlet from the handout). Unfortunately, this kind of situation, which probably will be the most frequent, will not be valuable when deciding the class of the graph, because it does not convey any detail that might help when distinguishing between the instances of the two classes.
However, the shortest path kernel will not encounter the same problem of not being sufficiently exposed to the differences in representation between the instances of the two classes. When counting the number of shortest paths with the same lengths, there will be a significant difference between the two types of graphs. For example, the shortest path between the first node and the last node from a path graph with n nodes is equal to $n - 1$, whereas in a cycle graph with the same number of nodes, we do not have shortest paths with length higher than $\lfloor \frac{n}{2} \rfloor$). So the shortest paths lengths in the case of a cycle graph vary from 1 to $\lfloor \frac{n}{2} \rfloor$, whereas, for a path graph, they vary from 1 to $n - 1$. Thus, its ability to capture this difference may be sufficient for the shortest path kernel to distinguish between the instances of the two classes and this justifies the high accuracy score the shortest paths kernel achieves on this problem.
Therefore, the graphlet kernel doesn't seem suitable for this problem, as it performs poorly, whereas the shortest path kernel seems to fit seamlessly in this situation.

Method	Accuracy (%)
Shortest path kernel	**100.0**
Graphlet kernel	40.0

Table 1: Comparison between results achieved with the shortest path kernel and the graphlet kernel

References

[1] Nils M. Kriege, Christopher Morris, Anja Rey, and Christian Sohler. A property testing framework for the theoretical expressivity of graph kernels. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 2348–2354. AAAI Press, 2018.

[2] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.

[3] Shriphani Palakodetye. The Smallest Eigenvalues of a Graph Laplacian. `http://blog.shriphani.com/2015/04/06/the-smallest-eigenvalues-of-a-graph-laplacian/`, 2015. [Online].

[4] Veronika Strnadova-Neeley. Spectral Clustering. `https://sites.cs.ucsb.edu/~veronika/SpectralClustering.pdf`. [Online].