Unsupervised Learning (MT7050) - Project 2

Instructions: *This project consists of 2 tasks that should be solved individually. Unless it is specified in the task, you are free to use any programming package in this project.*

The solution should be submitted at the course webpage in a single .pdf file ***with your source code attached as appendices*** (*i.e., don't put the source code in the main text*). *Your source code should include clear comments and documentation to describe what you are evaluating.*

Terminologies and notations in this project are the same as those in the von Luxburg paper.

TASK 1 (Spectral clustering)

This task allows you to experience how spectral clustering handles clusters of arbitrary shapes that cannot be identified correctly by traditional methods, such as k-means, Gaussian mixture models, etc. This task uses the Star_Data.txt dataset. The 1st and 2nd columns in Star_Data.txt are the x- and y-coordinates, respectively.

Note: Since the spectral clustering algorithms (essentially a matrix diagonalization) are very simple, I suggest writing your own code to implement the spectral clustering algorithm since packages in R or python may use different algorithms from those of the von Luxburg paper.

A) For the Star_Data.txt dataset, construct the similarity graph using an appropriate method (i.e., ε-neighborhood, kNN, mutual kNN or fully connected). Explain why the chosen method, edge weights and parameters are appropriate for the analysis. (10p)

B) Using the result from Part A, implement the "unnormalized spectral clustering" algorithm in P.5 of the von Luxburg paper with $k = 2$ clusters, and plot the data points in the new representation $(y_i)_{i=1,\cdots,n}$ (see von Luxburg for the meaning of y_i). Discuss if k-means can now be used to separate the clusters in the new representation and explain why it works or not. (15p)

C) Using the result from Part A, repeat part B but with the "normalized spectral clustering" algorithm using L_{sym} in P.6 of von Luxburg paper. (15p)

D) Based on the results in Part B and C, argue which Laplacian (L or L_{sym}) may be more suitable for clustering problems. (10p)

TASK 2 (Manifold Learning)

This task allows you to experience how the graph Laplacian and the commute-time-distance (CTD) embedding discussed in the class perform manifold learning. This task uses the dataset Swiss_Roll.txt. The 1st, 2nd and 3rd column in Swiss_Roll.txt are the x-, y- and z-coordinate, respectively. The aim of the task is to see if you can "unfold" the Swiss roll by applying the CTD embedding appropriately.

Note: The Swiss_Roll.txt dataset contains 2000 points, meaning that the dimension of the Laplacian matrix is 2000 by 2000. If your computer cannot handle the diagonalization of such big matrix, or if you do not have experience in sparse spectral decomposition methods, you can randomly sample a subset of points, e.g. 700 points, from the dataset to perform this task.

Note: The graph construction and implementation of the CTD embedding are almost the same as the algorithms in Task 1 and so only minor modification of your code from Task 1 is needed.

A) For the Swiss_Roll.txt dataset, construct the similarity graph using an appropriate method (i.e., ε-neighborhood, kNN, mutual kNN or fully connected). Explain why the chosen method, edge weights and parameters are appropriate for the analysis. (5p)

B) Using the result from Part A, construct and plot a 2-dimensional CTD embedding in terms of the eigenvalues and eigenvectors of the "unnormalized Laplacian". In this part, you should decide which two eigenvectors best unfold the Swiss roll and explain why this is so. (20p)

C) Using the result from Part A, repeat part B in terms of the eigenvalues and eigenvectors of the "normalized Laplacian" L_{sym}. (20p)

D) Based on the results in Part B and C, argue which Laplacian (L or L_{sym}.) may be more suitable to unfold the Swiss roll. (5p)

Final Remark: In this project, I have not included the random walk Laplacian in the analysis intentionally. This is because the random walk Laplacian is not a symmetric matrix and therefore its left and right eigenvectors are not the same. This means that one needs to first understand if the left or right eigenvectors should be used for the embedding. The theory behind the left and right eigenvectors of the random walk Laplacian, that relates to the diffusion process on a Markov network, is actually quite interesting. However, it is beyond the scope of this course.