Assignment 9

Statistical Machine Learning

Sebastian Bordt / David Künstle / Martina Contisciani / Nicolò Ruggeri / Rabanus Derr / Prof. Ulrike von Luxburg

Summer term 2020 — due on **July 2nd at 14:00**

Exercise 1 (Spectral graph theory 3+1+3+1=8 points) In the following we consider the symmetric normalized graph Laplacian $L = D^{-1/2}(D - A)D^{-1/2}$, where A denotes the adjacency matrix of an undirected and unweighted graph and D the graph's degree matrix, for various special graphs.

(a) Let G be an undirected, unweighted, and connected graph. Let $0 = \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_n$ denote the increasingly sorted eigenvalues of L. Prove that

$$\lambda_n \leq 2.$$

(b) Let G be the complete graph on n vertices, that is, any two vertices are connected. Show that its Laplacian has eigenvalues 0 with multiplicity one and $n/(n-1)$ with multiplicity $n-1$.

(c) Let G be the complete bipartite graph on $n+m$ vertices, that is, every vertex $v \in \{v_1, \ldots, v_n\}$ is connected with every vertex $\tilde{v} \in \{v_{n+1}, \ldots, v_{n+m}\}$, but no two vertices in $\{v_1, \ldots, v_n\}$ and $\{v_{n+1}, \ldots, v_{n+m}\}$ are connected. Show that its Laplacian has eigenvalues 0 with multiplicity one, 1 with multiplicity $m+n-2$, and 2 with multiplicity one.

(d) Let G be the star on n vertices, that is, v_1 is connected with every vertex $v \in \{v_2, \ldots, v_n\}$ but no two vertices in $\{v_2, \ldots, v_n\}$ are connected. Show that its Laplacian has eigenvalues 0 with multiplicity one, 1 with multiplicity $n-2$, and 2 with multiplicity one.

Hint: One way to construct the eigenvectors is to build the characteristic polynomial via induction.

Exercise 2 (Implementing k-means and spectral clustering, 2+1+3+1=7 points)

(a) Implement k-means in the function `kmeans_cluster(X, k)` which takes as input data points X_i with $i = 1, \ldots, n$ and parameter k. It computes the clustering and returns a vector of length n in which the i-th entry is from $\{0, 1, \ldots, k-1\}$ representing the cluster membership of the i-th data point. It also returns the cluster centers.

Points for this task will be granted for a correct implementation. In the next tasks you can use the built-in function `sklearn.cluster.KMeans` if you are not confident in your own implementation.

(b) In X you find a $n \times 2$ matrix with $n = 400$ samples in $\mathbb{R}^2$ from three clusters. Apply k-means with $k = 3$, plot the data and color the points according to their assigned cluster. Mark the cluster centers.

Repeat this for $k = 2$ and $k = 4$.

(c) Implement a function `kneighbors_graph(X,k)` which takes as input the data points (X_i) with $i = 1, \ldots, n$ and parameter k. Return the $n \times n$ weight matrix of the corresponding kNN-graph W where $W_{i,j} = 1$ if X_i is among the k nearest neighbors of X_j or vice versa. Otherwise set $W_{i,j} = 0$.

Implement a function `spectral_cluster(W,k,normalize)` which takes as input a kNN-graph, a parameter k for the number of clusters (not for the kNN-graph) and a boolean value indicating whether to use the normalized or unnormalized Laplacian. Perform spectral clustering, run k-means on the spectral embeddings and return the the centers, the clustering as well as the embedded data points as a $n \times k$ matrix.

In the next task you can use the built-in function `sklearn.cluster.SpectralCluster` if you are not confident in your own implementation.

(d) Apply unnormalized spectral clustering to the data set from task (b). Choose a reasonable parameter k for the kNN-graph and try to detect three clusters.

For the resulting clustering make the same plot as in (b) to visualize the cluster membership of each point. Visualize the spectral embedding in a 3-d scatter plot.

Repeat this with normalized spectral clustering.

Exercise 3 (Similarity graphs, 2+2+1=5 points)

Many machine learning algorithms build on graphs. In this exercise we explore similarity graphs, a simple method to transform data into graph representation.

(a) Implement the function `plot_similarities(X, sigma)`. It takes a data matrix X as input and computes the paired similarities according to the Gaussian kernel with parameter σ

$$s(x_i, x_j) = \exp(-||x_i - x_j||^2/(2\sigma^2)).$$

The similarities are then visualized in a heat map. In the interactive simulation, play with the kernel parameter σ on the *original two moons* and the *noisy two moons* dataset. Which values of `sigma` would you choose, and why? What is the role of noise, how does it influence your choice of `sigma`?

(b) Now implement the function `plot_graph(X, k, dim, mutual)`, which uses `kneighbors_graph` of `sklearn.neighbors` to create a k-nearest neighbor graph and and plot its adjacency matrix. Only the first `dim` dimensions of the input data `X` should be used to create the graph. Based on the parameter `mutual`, the graph should be either symmetric or mutual.

In the interactive simulation, play with the parameters `k` and `dim` on the *unbalanced Gaussian* data set. For $d = 2$, what is the smallest `k` for which the graph consists of a single connected component? How do both graphs look like for small and high values of `k`? Now start to increase the dimension. What happens for $d = 200$? Do you have a (theoretical) explanation for this effect?

(c) Now implement the function `plot_degree(X, k)`, which again creates a k-nearest neighbor graph and plots the degree of vertices in the graph.

In the interactive simulation, play with the parameter `k` and see how the vertex degrees change. For a suitable choice of `k`, there is a relation between vertex degrees in the k-nearest neighbor graph and the probability distribution from which the data is drawn. Can you guess what it is?