Clustering Analysis*

Ke Chen

To carry out this assessed coursework, you will need the Python notebook, `clustering.ipynb`, which contains code to get you started with each of the assignments, along with several data files required by different assignments (see the related assignments for details of those data sets) in this coursework. All above are available in a zipped file on BlackBoard alongside this document.

Clustering analysis in its own right is a classical topic in unsupervised learning to learn a high-level summary of data, which provides a representation of data from a global perspective. In this coursework, you are asked to implement state-of-the-art clustering algorithms in Python and apply your implementations along with the provided Python implementation of clustering algorithms to several meaningful synthetic datasets for clustering analysis.

1 K-means Clustering Analysis

K-means is one of the most commonly used clustering analysis algorithm. In this work, you are asked to apply the provided K-means functions to synthetic datasets to understand its behaviour and how to use a cluster validity index to decide the proper number of clusters underlying a data set.

Assignment 1 [3 marks] Apply the K-means built-in function, `sklearn.cluster.KMeans`, in the `scikit-learn` library with Euclidean distance to the 2-D dataset of 3 clusters, `kmeans_data_1.npy`, with three different initial conditions given in the notebook, `clustering.ipynb`. In your answer notebook, (a) implement a display function, `visualize_kmeans_res`, to enable you to display a partition yielded by K-means clustering where different clusters must be marked in different colours, and (b) use your implemented function to display the final partitions where the initial and final mean points of 3 clusters must be marked in the plot for **each** of 3 initialisation scenarios stated in the notebook, `clustering.ipynb`, and 6 plots with the proper titles must be arranged in two rows (one for initial means and the other for final means).

Assignment 2 [3 marks] K-means algorithm cannot be used until the hyperparameter K (the number of clusters) is set up. In a real application, however, the number of clusters is often unknown. In this circumstance, the scatter-based **F-ratio index**[1] may be applied to decide the number of clusters. In your answer notebook, (a) implement the scatter-based F-ratio index in Python where Euclidean distance is used; (b) for $K = 2, 3, \cdots, 10$, run the the K-means built-in function, `sklearn.cluster.KMeans`, in the `scikit-learn` library with Euclidean distance on the dataset, `kmeans_data_2.npy` (for each K, you must run K-means with **3** different random initialisation conditions set by yourself), then plot F-ratio index (y-axis) versus K (x-axis) (in this plot, for each

*Assessed Coursework: the deadline and requirements can be found at the end of this document.

[1] For the definition of the F-ratio index, see "K-means Clustering" lecture note.

K, use only the **least** F-ratio index value measured on 3 partitions caused by different initialisation conditions), and report the optimal number of clusters in this data set; and (c) display the final partition corresponding to the optimal number of clusters you find out in (b) with the display function you implemented in **Assignment 1**.

2 Spectral Clustering Analysis

As a state-of-the-art clustering analysis method, spectral clustering can deal with data of nonlinear and non-convex manifold with the connectivity criteria. In this work, you are asked to implement a spectral clustering algorithm in Python and apply your own implementation to synthetic datasets in order to understand how spectral clustering works on data sets in different nature.

Assignment 3 [4 marks] In your answer notebook, implement the **asymmetric normalised spectral clustering**[2] algorithm where the **fully connected graph** is used to generate the Laplacian matrix used in spectral clustering. You are asked to use two functions: `gaussian_similarity` for generating the fully connected graph and `asymmetric_SC` for the asymmetric normalised spectral clustering in your implementation. (**Hint**: To implement the `asymmetric_SC` function, you can use the built-in function, `np.linalg.eig`, in the `numpy` library for eigen analysis and the built-in function, `sklearn.cluster.KMeans`, in the `scikit-learn` library for K-means clustering. Also, you can use the toy data set given in the notebook to test your implementation.)

Assignment 4 [3 marks] Apply your implemented `gaussian_similarity` function in **Assignment 3** to the dataset, `SC_data_1.npy`, to generate a Laplacian matrix. In your answer notebook, (a) describe how you find out an appropriate hyperparameter value in the Gaussian kernel and explicitly report this value; (b) conduct eigen analysis on the Laplacian matrix, list all the elements of the eigenvector corresponding to the second smallest eigenvalue or the 1st smallest non-zero eigenvalue in a 4×10 table; and (c) by using the result achieved in (b), decide how many clusters in this data set and display clusters achieved in a 2-D plot where clusters are marked in different colours.

Assignment 5 [3 marks] Apply your implemented `gaussian_similarity` and `asymmetric_SC` functions in **Assignment 3** to the dataset, `SC_data_1.npy`, with the number of clusters you find out in **Assignment 4**, and two other datasets of two clusters, `SC_data_2.npy` and `SC_data_3.npy`, respectively. In your answer notebook, display clusters achieved in a 2-D plot where clusters are marked in different colours, respectively, for **all** three data sets.

3 Hierarchical Clustering Analysis

As one of the most important clustering analysis techniques, hierarchical clustering enables one to achieve all possible clusters at different levels for a data set. In this work, you are asked to apply the commonly-used agglomerative algorithm with different cluster-distance measures to a synthetic

[2] For the algorithmic description of this algorithm, see "Spectral Clustering" lecture note.

dataset in order to understand how this algorithm works.

Assignment 6 [3 marks] Choosing Euclidean distance to produce the initial distance matrix, apply the built-in function, `scipy.cluster.hierarchy` with three different cluster-distance measures[3], single-linkage, complete-linkage and group-average, in the `scipy` library to the dataset, `HC_data.npy`, respectively. In your answer notebook, (a) plot three dendrogram trees achieved by the use of three cluster-distance measures in the agglomerative algorithm; (b) report the number of clusters found by using the longest K-cluster lifetime criterion achieved from (a); and (c) display 3 partitions achieved in (b), where 3 plots corresponding to single-linkage, complete-linkage and group-average must be arranged in one row with the proper titles.

4 Ensemble Clustering Analysis

As one of the state-of-the-art clustering analysis techniques, ensemble clustering algorithms allow for combining different types of clustering algorithms and working on various feature sets of a data set to reach a synergy for clustering analysis. In this work, you are asked to implement the evidence-accumulated clustering algorithm and apply your implementation to two benchmark data sets to understand how this algorithm works.

Assignment 7 [2 marks] In your answer notebook, implement a function, `ensemble_clustering`, for the evidence-accumulated clustering algorithm[4] based on K-means and the agglomerative algorithms in Python. (**Hint**: To implement the `ensemble_clustering` function, you can use the built-in functions, `sklearn.cluster.KMeans`, in the `scikit-learn` library for K-means clustering, the `scipy.cluster.hierarchy.linkage` and `scipy.spatial.distance.squareform` in the `scipy` library for hierarchical clustering.)

Assignment 8 [4 marks] Apply your implemented `ensemble_clustering` function to two datasets, `SC_data_2.npy` and `SC_data_3.npy`, respectively. In your experiment, you need to decide how many initial partitions are generated by K-means algorithm with Euclidean distance and what is an appropriate cluster-distance measure to make the clustering ensemble algorithm work properly, In your answer notebook, (a) describe what are initial partitions and what the cluster-distance measure you used for **each** of two datasets, which lead to satisfactory results, and (b) plot two dendrogram trees achieved on two datasets and display clusters in **each** partition of two datasets in a 2-D plot where clusters are marked in different colours.

Requirement: Before starting working on this assessed coursework, you need to

1. download all the files required by this coursework from Blackboard as specified at the beginning of this document;
2. unzip the file then you should be see a Jupyter notebook file named `clustering.ipynb` and one sub-directory named `Data` (you must keep this directory/sub-directory structure and its name unchanged when you work on this coursework);

[3] For the definition of cluster-distance measures, see "Hierarchical Clustering" lecture note.
[4] For the algorithmic description of this algorithm, see "Clustering Ensemble" lecture note.

3. rename `clustering.ipynb` in the directory as `yourfullname_clustering.ipynb`. For instance, if your name is "John Smith", your filename should be `john_smith_clustering.ipynb`. This file will be your answer notebook to be submitted for marking, so you must include everything required by the coursework in this Jupyter notebook.

Deliverable: Only your answer notebook, `yourfullname_clustering.ipynb`, which should include all your code, output, answers and your interpretation/justification. In this Jupyter notebook, all assignments have been separated with the clear delimiters. You must put your stuff regarding an assignment in those cells related to this assignment and, if necessary, create new cells within the delimiters of this assignment.

Your answer notebook, `yourfullname_clustering.ipynb`, must be submitted via the Blackboard.

Marking: Marking will be on the basis of (1) correctness of results and quality of comments on your code; (2) rigorous experimentation; (3) how informative and clear your results and answers are presented in your answer book; and (4) your knowledge exhibited, interpretation and justification.

Late Submission Policy: The default university late submission policy (for details, see the official document: http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=29825) is applied to this coursework.

Extension Policy: The default departmental extension policy is applied to this coursework. That is, no extension is allowed unless you have a mitigating circumstance. If you have any mitigating circumstance and want to make an extension, you should submit the completed mitigating circumstance form to SSO (for details, see the departmental Mitigating Circumstances page: http://studentnet.cs.manchester.ac.uk/assessment/mitigatingcircumstances.php?view=pgt). Note: the decision will be made by the departmental mitigating circumstance panel rather than the lecturer.

Deadline: 23:30, 26th January 2021 (Tuesday)