Norwegian University of Science and Technology

TDT4300 — **Solution to Assignment 3**

CLUSTERING

tdt4300-undass@idi.ntnu.no

Spring 2023

1 k-Means Clustering

1.1 Assignment

This is a programming part of the assignment. Your task is to implement the k-means clustering algorithm and assess the quality of the outputs by calculating Silhouette Coefficient. You are given a Jupyter Notebook[1] (formerly known as the IPython Notebook) file `k_means_clustering.ipynb` in which you have to implement only two functions: `kmeans()` and `silhouette_score()`. Everything else has already been prepared for you. As you have maybe already guessed, the programming language of our choice is Python.

Before you start, you need to install Jupyter Notebook and Python 3. Having that done, open your terminal, navigate to a folder with the `k_means_clustering.ipynb` file, and execute the command `jupyter notebook`. A window of your Internet browser should pop-up with the Jupyter Notebook interface. Open the `k_means_clustering.ipynb` notebook, read it carefully through, and execute it line by line. If this is new to you, get yourself familiar with the Jupyter Notebook and Python.

The assignment is very easy as you do not have to worry about anything else except the core k-means algorithm and Silhouette Coefficient. If you consider yourself a good programmer but without knowledge of Python, you should not have struggles, and you can add a new programming language to your portfolio. If you consider yourself a rather unexperienced programmer and without knowledge of Python, it is a good chance to learn new beginner friendly programming language, and gain more practice in programming. If programming scares you, seek help from other students. Use Piazza to find help if you do not know anyone. We do not have to remind you that plagiarism is not tolerable.

[1] https://jupyter.org/

1.2 Solution

An example implementation of the k-means clustering algorithm will not be provided.

2 Hierarchical Agglomerative Clustering (HAC)

2.1 Assignment

(a) Explain the Hierarchical Agglomerative Clustering (HAC) and the difference between MIN-link and MAX-link.

(b) You are given a two-dimensional dataset shown in Table 1. Perform HAC (for both MIN-link and MAX-link) and present the results in the form of dendrogram. Use the Euclidean distance. **Describe thoroughly the process and the outcome of each step.**

(c) Verify your results using the KNIME data analytics platform. For clarification, MIN-link and MAX-link is in KNIME referred as *SINGLE* and *COMPLETE* linkage methods. We provide you the file *data_hac.csv* containing the very same data. **Present a picture of your workflow and the dendrograms.**

ID	x	y
A	5	4
B	4	7
C	6	8
D	8	2
E	12	7
F	11	6

Table 1: Dataset for HAC.

2.2 Solution

(a) Hierarchical Agglomerative Clustering is described in detail in Section 8.3 of "Introduction to Data Mining" by Tan at al. The clustering algorithm starts with individual data points as the initial clusters and in each step merges the closest clusters. The proximity of two clusters can be defined in several ways; MIN-link and MAX-link are two of them.

MIN-link, MAX-link. From the Section 8.3.1 of the book: Min-link (single-link) defines cluster proximity as the proximity between the closest two points that are in different clusters, or using graph terms, the shortest edge between two nodes in different subsets of nodes. Max-link (complete link) takes the proximity between the farthest two points or using graph terms, the longest edge between two nodes in different subsets of nodes.

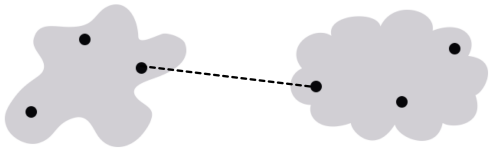

Figure 1: MIN-link

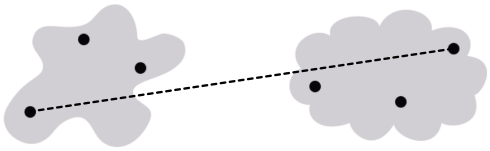

Figure 2: MAX-link

(b) Using the Euclidean distance, we first calculate the distance matrix for the initial data points:

d	A	B	C	D	E	F
A	0	-	-	-	-	-
B	3.16	0	-	-	-	-
C	4.12	2.24	0	-	-	-
D	3.61	6.40	6.32	0	-	-
E	7.62	8	6.08	6.40	0	-
F	6.32	7.07	5.38	5	**1.41**	0

Table 2: Distance matrix (E and F will be merged into a cluster).

MIN-link We see that E and F are the closest points and therefor they form a new cluster. In the MIN-link version, we select one of the cluster's points so that the distance to the other points is shortest possible. Following tables illustrate present result after each iteration of the algorithm.

d	A	B	C	D	E,F
A	0	-	-	-	-
B	3.16	0	-	-	-
C	4.12	**2.24**	0	-	-
D	3.61	6.40	6.32	0	-
E,F	6.32	7.07	5.38	5	0

Table 3: Distance matrix (B and C will be merged into a cluster).

d	A	B,C	D	E,F
A	0	-	-	-
B,C	**3.16**	0	-	-
D	3.61	6.40	0	-
E,F	6.32	7.07	5	0

Table 4: Distance matrix (A and B,C will be merged into a cluster).

d	A,B,C	D	E,F
A,B,C	0	-	-
D	**3.61**	0	-
E,F	6.32	5	0

Table 5: Distance matrix (D and A,B,C will be merged into a cluster).

d	A,B,C,D	E, F
A,B,C,D	0	-
E,F	5	0

Table 6: Final distance matrix.

MAX-link

In the MAX-link algorithm, we still select the closest points to form a cluster. The difference from the MIN-link is that when we recalculate the distance matrix, we keep the longest distance between a newly formed cluster and the rest of data points.

d	A	B	C	D	E,F
A	0	-	-	-	-
B	3.16	0	-	-	-
C	4.12	**2.24**	0	-	-
D	3.61	6.40	6.32	0	-
E,F	7.62	8	6.08	6.40	0

Table 7: Distance matrix (B and C will be merged into a cluster).

d	A	B,C	D	E,F
A	0	-	-	-
B,C	4.12	0	-	-
D	**3.61**	6.40	0	-
E,F	7.62	8	6.40	0

Table 8: Distance matrix (A and D will be merged into a cluster).

d	A,D	B,C	E,F
A,D	0	-	-
B,C	**6.40**	0	-
E,F	7.62	8	0

Table 9: Distance matrix (B,C and A,D will be merged into a cluster).

d	A,B,C,D	E,F
A,B,C,D	0	-
E,F	8	0

Table 10: Final distance matrix.

The resulting dendrograms are shown in Figures 3 and 4.

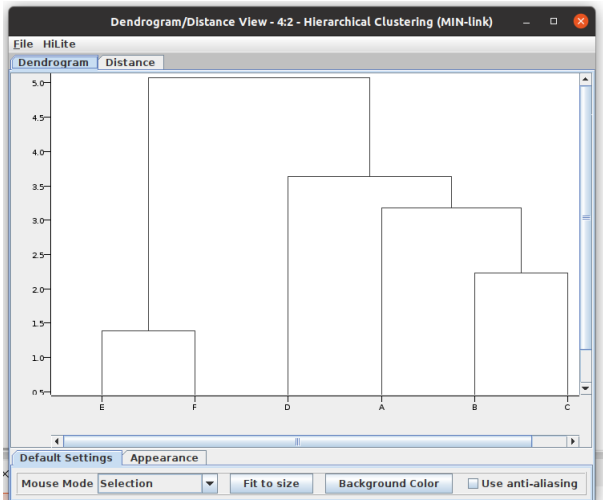

Figure 3: Dendrogram for MIN-link.

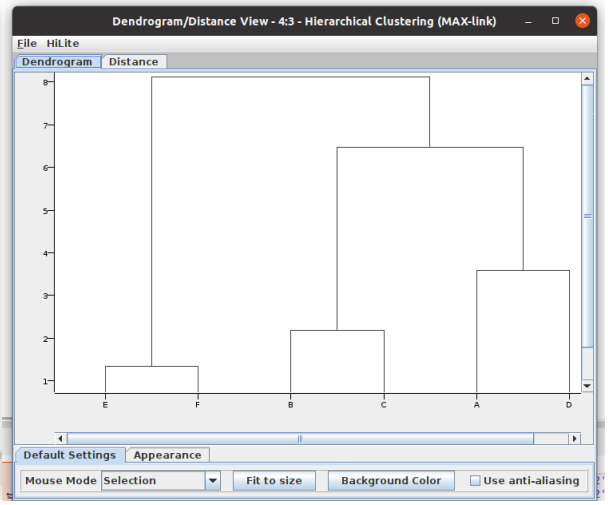

Figure 4: Dendrogram for MAX-link.

(c) Following figures show the Knime workflow and resulting dendograms.

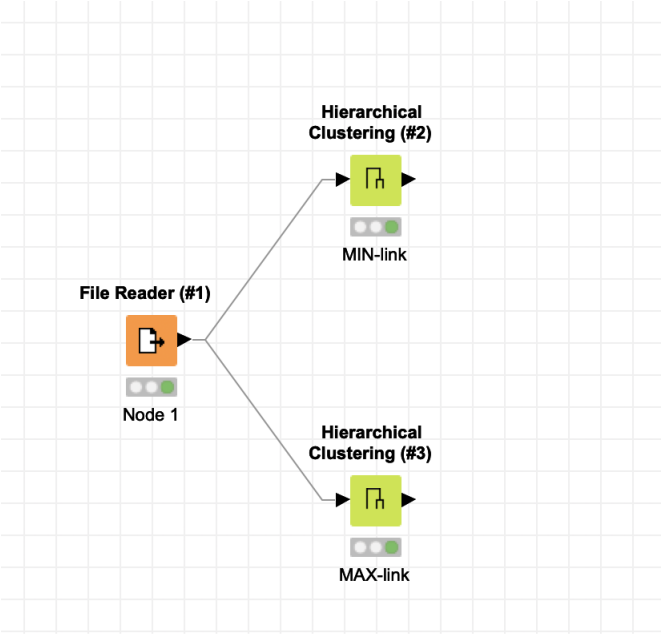

Figure 5: KNIME workflow.

3 DBSCAN Clustering

3.1 Assignment

You are given following points: $P_1 = (2,2)$, $P_2 = (13,7)$, $P_3 = (7,13)$, $P_4 = (2,1)$, $P_5 = (6,12)$, $P_6 = (12,5)$, $P_7 = (3,11)$, $P_8 = (13,9)$, $P_9 = (1,2)$, $P_{10} = (9,2)$, $P_{11} = (4,8)$, $P_{12} = (9,11)$, $P_{13} = (15,6)$, $P_{14} = (3,5)$, $P_{15} = (7,5)$, $P_{16} = (3,2)$, $P_{17} = (12,6)$, $P_{18} = (12,12)$.

(a) Your task is to perform DBSCAN clustering given the parameters $Eps = 3$ (Euclidean metric) and $MinPts = 2$ (including the analyzed point). Identify core, border and noise points. Identify clusters. **Describe thoroughly the process and the outcome of each step.**

(b) Verify your results using the KNIME data analytics platform. We provide you the file *data_dbscan.csv* containing the very same data. **Present a picture of your workflow and the scatter plot with marked clusters and outliers.**

Describe thoroughly the process and the outcome of each step.

3.2 Solution

First, we are going to label the points as core, border or noise points. For the labeling we need to know the neighborhood N of each point considering $Eps = 3$. The Euclidean distance between each point must be calculated in order to explore the neighborhood.

d^2	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}	P_{17}	P_{18}
P_1	0	146	146	**1**	116	109	82	170	**1**	49	40	130	185	10	34	**1**	116	200
P_2	146	0	72	157	74	**5**	116	**4**	169	41	82	32	**5**	104	40	125	**2**	26
P_3	146	72	0	169	**2**	89	20	52	157	125	34	**8**	113	80	64	137	74	26
P_4	**1**	157	169	0	137	116	101	185	**2**	50	53	149	194	17	41	**2**	125	221
P_5	116	74	**2**	137	0	85	10	58	125	109	20	10	117	58	50	109	72	36
P_6	109	**5**	89	116	85	0	117	17	130	18	73	45	10	81	25	90	**1**	49
P_7	82	116	20	101	10	117	0	104	85	117	10	36	169	36	52	81	106	82
P_8	170	**4**	52	185	58	17	104	0	193	65	82	20	13	116	52	149	10	10
P_9	**1**	169	157	**2**	125	130	85	193	0	64	45	145	212	13	45	**4**	137	221
P_{10}	49	41	125	50	109	18	117	65	64	0	61	81	52	45	13	36	25	109
P_{11}	40	82	34	53	20	73	10	82	45	61	0	34	125	10	18	37	68	80
P_{12}	130	32	**8**	149	10	45	36	20	145	81	34	0	61	72	40	117	34	10
P_{13}	185	**5**	113	194	117	10	169	13	212	52	125	61	0	145	65	160	**9**	45
P_{14}	10	104	80	17	58	81	36	116	13	45	10	72	145	0	16	**9**	82	130
P_{15}	34	40	64	41	50	25	52	52	45	13	18	40	65	16	0	25	26	74
P_{16}	**1**	125	137	**2**	109	90	81	149	**4**	36	37	117	160	**9**	25	0	97	181
P_{17}	116	**2**	74	125	72	**1**	106	10	137	25	68	34	**9**	82	26	97	0	36
P_{18}	200	26	26	221	36	49	82	10	221	109	80	10	45	130	74	181	36	0

Table 11: Distance matrix.

With $MinPts = 2$ (we understand it as the minimum count of the neighborhood points, including the analyzed point). The points P_1, P_2, P_3, P_4, P_5, P_6, P_8, P_9, P_{12}, P_{13}, P_{14}, P_{16} and P_{17} are labeled as the core points. The are no border points, since $MinPts = 2$. All other points (P_7, P_{10}, P_{11}, P_{15}, P_{18}) are the noise points which can be eliminated.

We define a cluster for each group of core points that are within Eps of each other and assign each border point to one of the cluster of its associated core points. This results in three clusters $C_1 = \{P_1,P_4,P_9,P_{14},P_{16}\}$, $C_2 = \{P_2,P_6,P_8,P_{13},P_{17}\}$ and $C_3 = \{P_3,P_5,P_{12}\}$.

It is not really necessary to calculate the distance matrix in this exercise. The dataset is very small and illustrating it in a Cartesian coordinate plane can reveal the distances between the points.

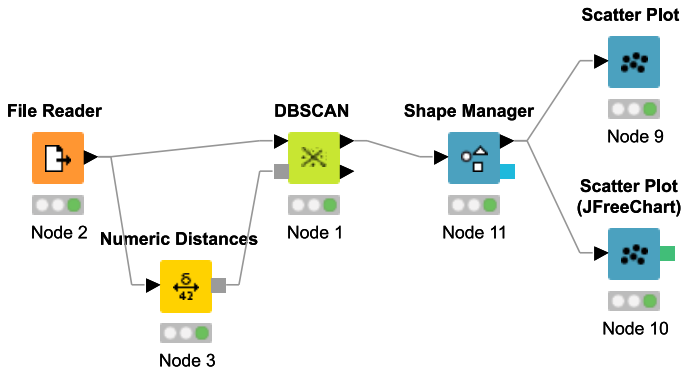

Figure 6: KNIME workflow.

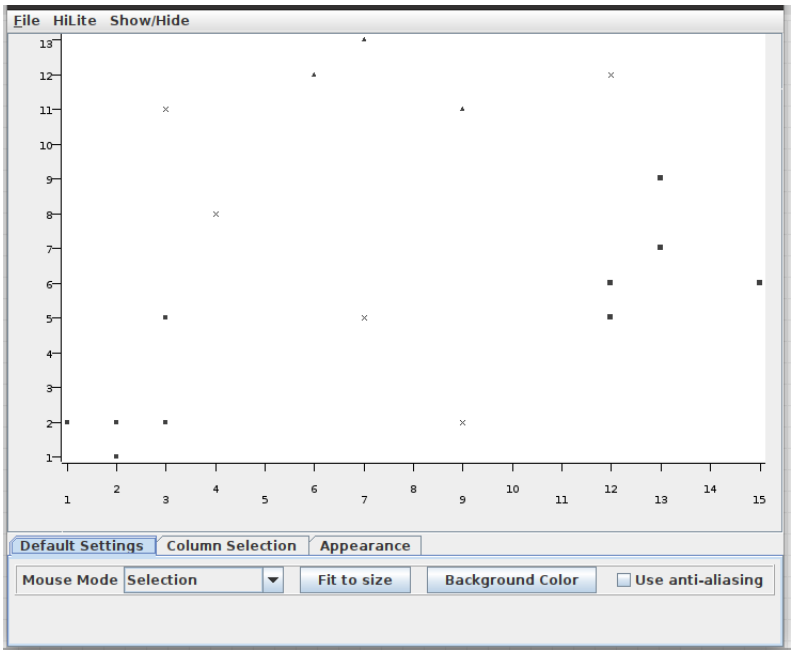

Figure 7: KNIME generated plot.