TDT4300 - Data Warehousing and Data Mining

Assignment 3

Martin Johannes Nilsen

22nd February 2022

1 K-Means Clustering

In this task we were to implement two functions in the Jupyter Notebook provided, one for K-Means and one for calulcating the silhouette score. The two functions were implemented using both standard Python primitives and Numpy arrays where I found it to be appropriate. My main goal was to implement the functions using my own logic, explaining each step I perform thoroughly. This is done both for you to follow my stream of thoughts, but also for my own understanding. The Jupyter Notebook file is delivered in both the *.ipynb* and *.html* format as requested.

2 Hierarchical Agglomerative Clustering (HAC)

2.1 Task 2a - HAC and MIN/MAX-link

As lectured in the subject, there are two types of hierarchical clustering; divisive and agglomerative clustering. In Hierarchical Agglomerative Clustering, we start off by each point being in its own cluster. For each iteration, you merge the closest pair of objects/points into one cluster, until you reach a desired amount of clusters (if supervised, else you will need to measure the performance for stop criteria, where cross validation may be one solution).

Another important factor when it comes to clustering, is the measure of distance between clusters. Two types of measurements for cluster distance is MIN and MAX-links. The MIN-link distance uses the closest points in each cluster, giving the minimal distance between two clusters. The MAX-link on the other hand, as you may have guessed, uses the distance of the two points being the furthest apart. Which one of these measures you use as the unit of distance may have a large impact on the outcome of clusters.

2.2 Task 2b - Execution of HAC with both MIN and MAX

For this task, I started off by computing the Euclidean distance for each point. This can be done by either defining each point as tuples in Python and run *math.dist()*, or calculated by hand by the equation for Euclidean distance.

$$dist = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Yielding the following table

	A	B	C	D	E
A	0	-	-	-	-
B	4.12	0	-	-	-
C	4.12	7.07	0	-	-
D	1.0	3.16	4.47	0	-
E	7.21	7.0	5.39	6.71	0

Table 1: Table of euclidean distances between clusters

MIN-link:

As we are using MIN-link, it is given that we select the lowest distance between the points in the newly formed cluster and the other clusters, when updating the table of distances.

Step 1 Merge AD as this is the shortest distance. Now we have to recalculate the distances from AD to the other clusters. We use the lowest distance to each of the other points from either A or D.

Step 2 Merge ADB

Step 3 Merge ADBC

Step 4 Merge ADBCE. Now we have one large cluster.

	AD	B	C	E
AD	0	-	-	-
B	3.16	0	-	-
C	4.12	7.07	0	-
E	6.708	7.0	5.39	0

(a) Step 1

	ADB	C	E
ADB	0	-	-
C	4.12	0	-
E	6.708	5.39	0

(b) Step 2

	ADBC	E
ADBC	0	-
E	5.39	0

(c) Step 3

Figure 1: Distances for each step in HAC with MIN-link

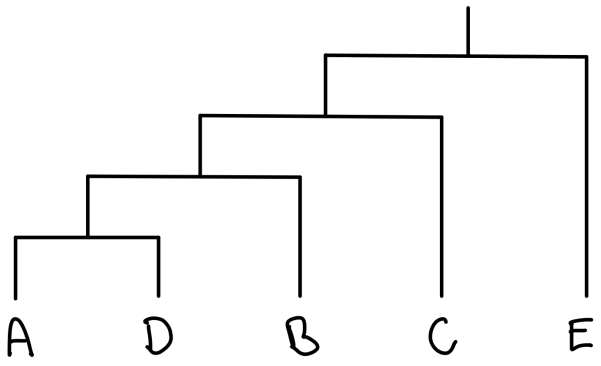


Figure 2: Dendogram for HAC with MIN-link

MAX-link:

As with the HAC using MIN-link, we perform the exact same steps with MAX-link, except that when we merge the closest clusters, we set the new distance to other clusters as the the distance of the points being the furthest apart.

Step 1 Merge AD and use the largest distance from the newly generated cluster to the others when updating the table.

Step 2 Merge ADB

Step 3 Merge CE

Step 4 Merge ADBCE

	AD	B	C	E
AD	0	-	-	-
B	4.12	0	-	-
C	4.47	7.07	0	-
E	7.21	7.0	5.39	0

(a) Step 1

	ADB	C	E
ADB	0	-	-
C	7.07	0	-
E	7.21	5.39	0

(b) Step 2

	ADB	CE
ADB	0	-
CE	7.21	0

(c) Step 3

Figure 3: Distances for each step in HAC with MAX-link

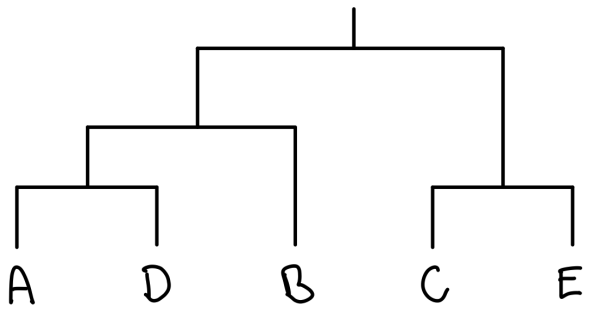
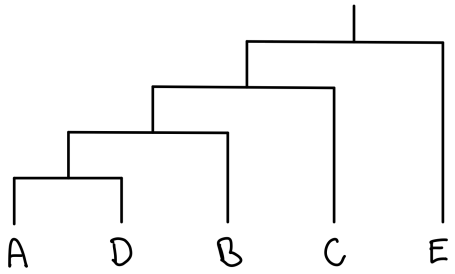


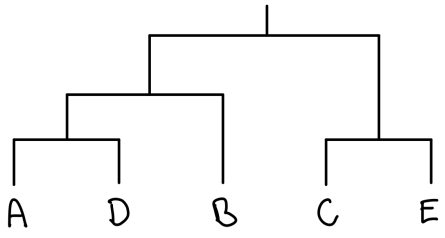
Figure 4: Dendogram for HAC with MAX-link

2.3 Task 2c - Verification of results

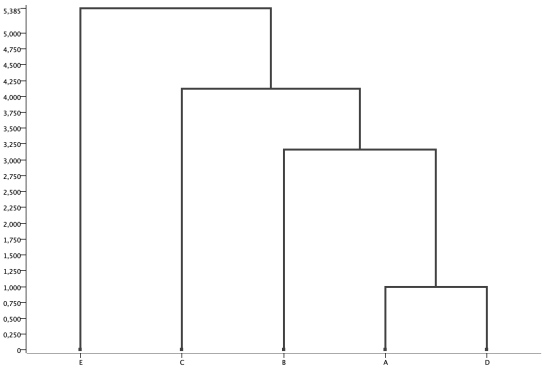
In the final task I sat up a KNIME workflow for verifying the results in the former subtask. As you can observe in Figure 5, the dendograms by hand seems to be the same as the dendograms created by the KNIME-nodes. The only difference seems to be that KNIME includes the distances along the y-axis in the figure, while I have only illustrated the lines indicating merges. An image of the KNIME-workflow is added below.



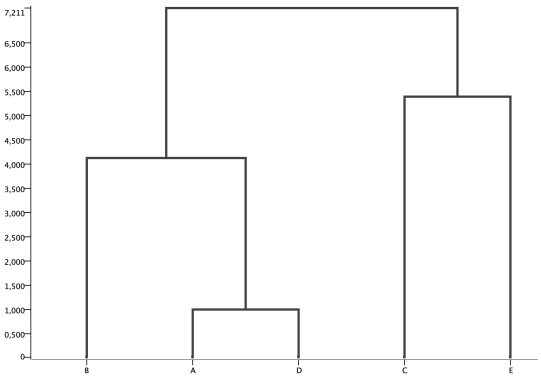
Dendogram with MIN-link by hand



Dendogram with MAX-link by hand



Dendogram with MIN-link in KNIME



Dendogram with MAX-link in KNIME

Figure 5: Dendogram comparison of HAC by hand vs in KNIME

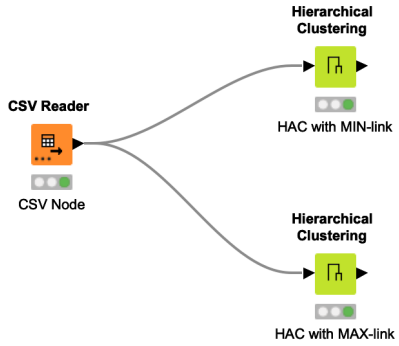


Figure 6: KNIME workflow

3 DBSCAN Clustering

3.1 Task 3a - DBSCAN by hand

For solving this task I chose to utilize Geogebra for plotting all the points in a graph, making it easier to observe the characteristics we are looking for. Note that I could have used the same method as in the former task, using a correlation matrix with distances and checked for distances below/above $minPts$.

The first step is to plot all points, and create circles around each with the radius equal to $Eps = 4$. We are tasked to identify core, border and noise points. I use the following definitions for identifying these characteristics:

- A **core** point has more than or equal amount of points as the given $minPts$ in its radius. Note that the defined $minPts$ includes the analyzed point.
- A **border** point has less points than the given $minPts$ in its radius, but is in the neighborhood of a core point.
- A **noise** point are none of the above.

After having plotted the points and utilized the function $Circle(point, radius)$ for drawing a circle around them, we can continue to characterize each point. The core points have 3 or more points in its radius (including itself), and we end up with the following core points: $(P_3, P_4, P_8, P_9, P_{10}, P_{11}, P_{14})$. The noise points are the points not inside a radius, which yields: $(P_0, P_1, P_2, P_6, P_7)$. The remaining points then have to be border points, being the points: (P_5, P_{12}, P_{13}). This can be confirmed by checking if the border points are all inside a radius of a core point in Figure 7, which they are. As we can see in the figure we end up with 2 clusters.

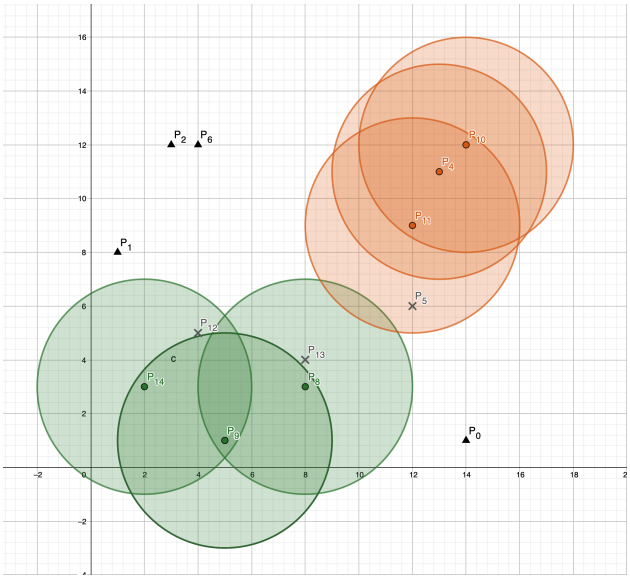


Figure 7: Clusters

3.2 Task 3b - Verification of results

In this task I used KNIME for verifying the results from the former subtask. Using the workflow in Figure 8,

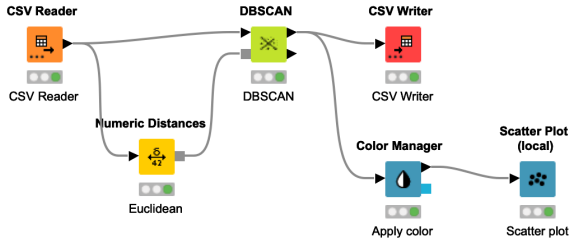


Figure 8: KNIME workflow

I was able to get the following output of the KNIME workflow, in the form of a csv table and a scatter plot. Please note that I have transposed the table to horisontal layout, and replaced the coordinates with a point label instead.

Point	P_0	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
Cluster	Noise	Noise	Noise	Cluster_1	Cluster_0	Cluster_0	Noise	Noise	Cluster_1	Cluster_1	Cluster_0	Cluster_0	Cluster_1	Cluster_1	Cluster_1

Figure 9: Output table from KNIME

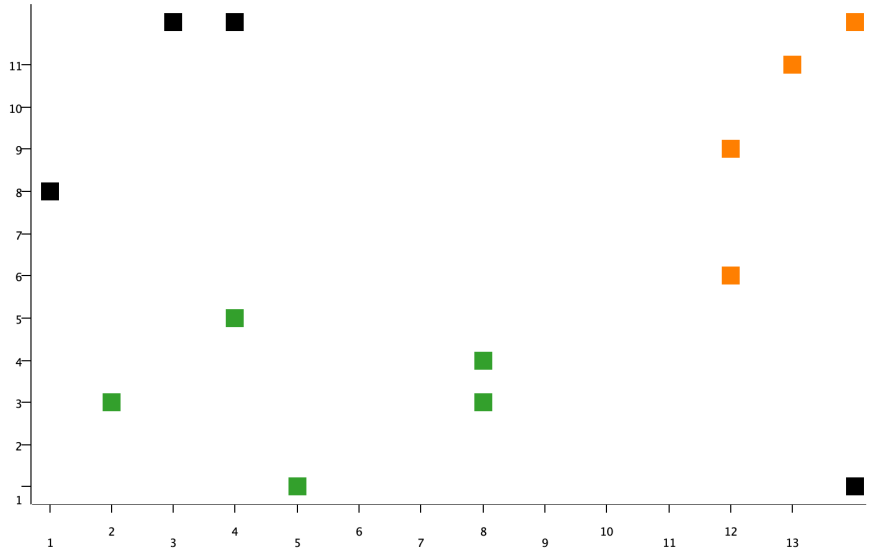


Figure 10: Scatter plot from KNIME

And we can see that the KNIME workflow yields the same result as in task 3a. For reference, I selected the same colors in Figure 7 and 10. As you may observe, cluster 0 is the orange one, being the same in both figures, and the same for the green cluster being cluster 1. The only difference is that the border and core points are differentiated in both color and shape in my plot, which the scatter plot from KNIME does not.