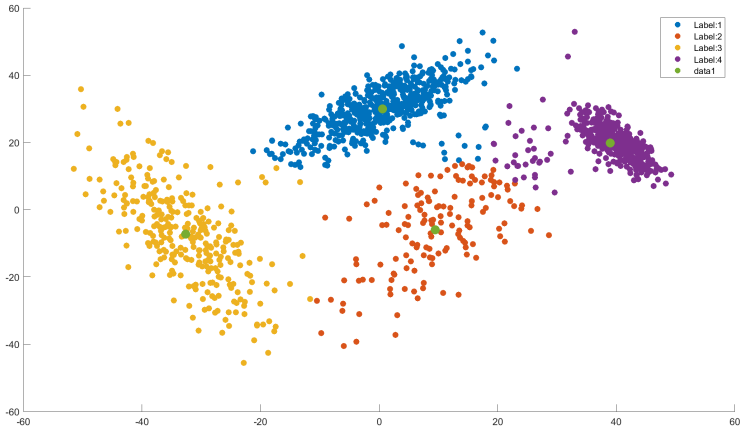
Home Assignment 1 - Performance Evaluation

Allar Viinamäe - 163578IAPM

Representative based clustering using k-means algorithm

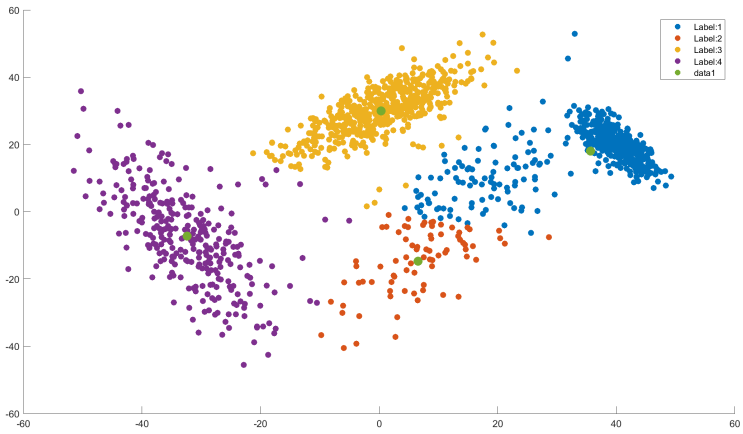
Table 1: Hyper parameters for k-means

Parameter	Value
clusters	4
iterations	10
metric function	Euclidean distance



Result with own implementation using Euclidean distance. Here green marks represent final position of centroids. Centroids have almost successfully converged into 5 clusters in 10 iterations. Still, the 1th and 4th clusters shouldn't ideally count some of the points of 2th cluster as their points, but this is a specific of k-means algorithm, since it uses the mean of the cluster to set it's cluster range and can't explicitly distinguish "border points". Execution time is 0.514800 seconds.

Result with Matlab's implementation. Matlab's implementation produces identical result. The only difference is in the execution time. Own implementation could be improved by identifying the convergence of the algorithm faster, since the convergence actually happens in less than 10 iterations. 10 iterations was chosen randomly and with the goal to definitely make the algorithm converge. Execution time is 0.019262 seconds.



Result with own implementation using Cosine distance. Euclidean distance seems to best fit the k-means in general, but here Cosine distance is used for comparison. I think this is a result of the fact that k-means works by measuring distance from a dataset point to a centroid and the distance is different with Cosine distance.

Density based clustering using DBSCAN

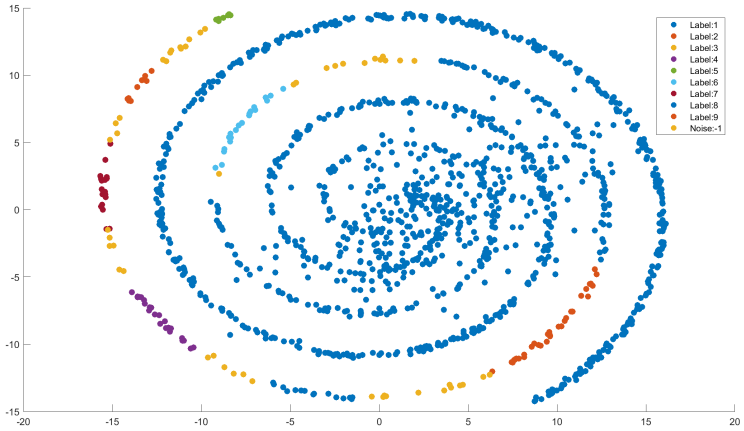
Table 2: Hyper parameters for DBSCAN on regular dataset

Parameter	Value
epsilon	4
minPts	7
metric function	Euclidean distance

Result with own implementation using Euclidean distance. Points labeled as -1 represent noise points. Here a rather faulty extra small yellow colored cluster is labeled. This is a result of "tightly" grouped 7 data points labeled as core points and they also they seem to be further away from a nearby bigger cluster labeled as 1.

Table 3: Hyper parameters for DBSCAN on spiral dataset

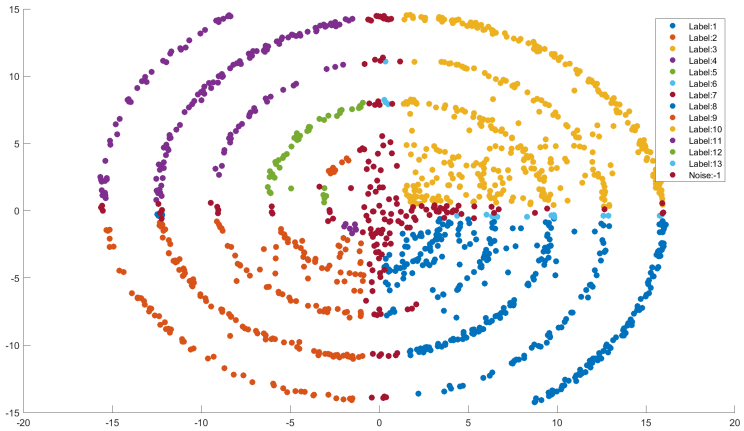
Parameter	Value
epsilon	1.5
minPts	8
metric function	Euclidean distance



A dataset in spiral form is tested. Very hard to distinguish spiral cluster from the second gaussian distributed cluster. Note that decreasing minPts produces greater amount of different clusters. Execution time is 72.369003 seconds.

Table 4: Hyper parameters for DBSCAN on spiral dataset using Canberra distance

Parameter	Value
epsilon	0.2
minPts	4
metric function	Canberra distance



For comparison spiral dataset is tested with Canberra distance function. Clear sector clusters

have been produced, but clearly not the point of DBSCAN clustering algorithm. Execution time is 23.882735 seconds.

Conclusion

For k-means algorithm, own implementation and Matlab's implementation give identical results, but Matlab's code is more efficient regarding execution time. Compared to DBSCAN, k-means runs much faster and doesn't require knowledge about cluster count prior to executing the algorithm. DBSCAN runs a lot longer, but can produce clusters in arbitrary shapes and also labels points not very close to defined clusters as noise, which can be useful to better distinguish clusters.