Assignment 6: Report

Clustering

Danny Rogaar (s2393344)
Daan Opheikens (s3038416)
Panagiotis Giagkoulas (s3423883)
Saim Eser Comak (s3432548)
Carlos Huerta (s3743071)
Emile Muller (s3787915)

Group 14

October 14, 2018

1 Pen-and-paper question

1.1 Single Linkage Clustering

	p1	p2	p3	p4	p5
p1	0.0	0.9	0.6	0.5	0.7
p2	0.9	0.0	0.4	0.6	**0.1**
p3	0.6	0.4	0.0	0.6	0.2
p4	0.5	0.6	0.6	0.0	0.3
p5	0.7	**0.1**	0.2	0.3	0.0

Table 1: Original distance matrix

The first 2 closest clusters are p2 and p5 (**0.1**).
Merging them creates a new cluster **p2-p5**, with the shortest distance from p2 and p5.

	p1	p3	p4	p2-p5
p1	0.0	0.6	0.5	0.7
p3	0.6	0.0	0.6	**0.2**
p4	0.5	0.6	0.0	0.3
p2-p5	0.7	**0.2**	0.3	0.0

The next 2 closest clusters are p3 and p2-p5 (**0.2**).
Merging them creates a new cluster **p2-p3-p5**, with the shortest distance from p3 and p2-p5.

	p1	p4	p2-p3-p5
p1	0.0	0.5	0.6
p4	0.5	0.0	**0.3**
p2-p3-p5	0.6	**0.3**	0.0

The next 2 closest clusters are p4 and p2-p3-p5 (**0.3**).
Merging them creates a new cluster **p2-p3-p4-p5**, with the shortest distance from p4 and p2-p3-p5.

	p1	p2-p3-p4-p5
p1	0.0	**0.5**
p2-p3-p4-p5	**0.5**	0.0

The next 2 closest clusters are p1 and p2-p3-p4-p5 (**0.5**).

Merging them creates a new cluster **p1-p2-p3-p4-p5**, with the shortest distance from p1 and p2-p3-p4-p5. This gives us the final cluster.

	p1-p2-p3-p4-p5
p1-p2-p3-p4-p5	0.0

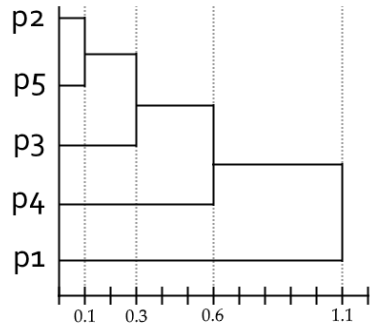

Figure 1: Dendrogram using the Single Linkage Clustering

1.2 Average Linkage Clustering - UPGMA Method

Now we will employ Unweighted Pair Group Method with Arithmetic Mean (UPGMA) Method on the same distance matrix in table 1. Using the following equation:

$$\frac{1}{|c_1|\,|c_2|}\sum x_1 \epsilon c_1 \sum x_2 \epsilon c_2 D(x_1, x_2) \tag{1}$$

Where c_1 and c_2 are the clusters in which $|c_1|$ and $|c_2|$ is the notation for the cardinality of these clusters, we select points originating from each potential cluster and calculate the mean distance between each points, D(x_1,x_2). This process is iterated and with each iteration the distance matrix is updated until all clusters are formed which we then structure these clusters into a dendrogram tree diagram.

We begin by finding the shortest distance between two points. In this case the D($p5$,$p2$) seems to be the shortest distance. We can imagine these points placed around a cluster center with a total of 0.1 Euclidean distance units between each point. Then the average of this distance would give us how far these points are away from the cluster center. Forming our first cluster, our points are $0.1/2 = 0.05$ Euclidean distance units away from the first cluster center. Next we calculate how far the newly created cluster is away from remaining points (p1,p3,p4). Each distance calculation is followed by taking the mean of outputted distanced.

$$\begin{aligned} D((p5,p2),p1) &= (D(p5,p1) + D(p2,p1))/2 \\ &= (0.7 + 0.9)/2 = 0.8 \\ D((p5,p2),p3) &= (D(p5,p3) + D(p2,p3))/2 \\ &= (0.2 + 0.4)/2 = 0.3 \\ D((p5,p2),p4) &= (D(p5,p4) + D(p2,p4))/2 \\ &= (0.3 + 0.6)/2 = 0.45 \end{aligned}$$

After calculating new distances, we update our distance matrix with the newly calculated clusters. Looking at table 2, notice that the columns and rows designated for p2 and p5 are removed since the new cluster between $p5$ and $p2$

	(p5,p2)	p1	p3	p4
(p5,p2)	0.0	**0.8**	**0.3**	**0.45**
p1	**0.8**	0.0	0.6	0.5
p3	**0.3**	0.6	0.0	0.6
p4	**0.45**	0.5	0.6	0.0

Table 2: Updated distance matrix after first iteration

is formed. Again, we select the shortest distance between two points. In this case the shortest distance can be found between cluster ($p5$,$p2$) and p3 which is 0.3 Euclidean distance units. Next we calculate the distance between cluster (($p5$,$p2$),$p3$) and rest of the points, namely p1 and p4. Following computations are calculated.

$$\begin{aligned} D(((p5,p2),p3),p1) &= (D((p5,p2),p1) * 2 + D((p3,p1))/3 \\ &= ((0.8) * 2 + 0.6)/3 = 0.73 \end{aligned}$$

$$\begin{aligned} D(((p5,p2),p3),p4) &= (D((p5,p2),p4) * 2 + D((p3,p4))/3 \\ &= ((0.45) * 2 + 0.6)/3 = 0.5 \end{aligned}$$

Notice that we used proportional averaging, accounting for the cardinality of two elements in cluster (p5,p2). Then we update the distance matrix

	((p5,p2),p3)	p1	p4
((p5,p2),p3)	0.0	**0.73**	**0.5**
p1	**0.73**	0.0	0.5
p4	**0.5**	0.5	0.6

Table 3: Updated distance matrix after second iteration

Again, we removed the columns and rows designated for (p5,p2) cluster and p3 point. Next we find the shortest distance between points. In this case distance between p4 and p1 is the shortest distance with the value of 0.5 Euclidean distance. Although the same distance is present between cluster ((p5,p2),p3) and p4, it doesn't matter which distance we take as the shortest one. After computing our cluster (p1,p4), we update the distance matrix one last time. Table 3

$$\begin{aligned} D((p1,p4),((p5,p2),p3),p1) &= (D(p1,((p5,p2),p3)) + D(p4,((p5,p2),p3)))/2 \\ &= (0.73 + 0.5)/2 = 0.615 \end{aligned}$$

	((p5,p2),p3)	(p1,p4)
((p5,p2),p3)	0.0	**0.615**
(p1,p4)	**0.615**	0.0

Table 4: Updated distance matrix after third iteration

Finally, we draw our dendrogram(figure 2) based on the shortest distances used to find clusters until now. They are of the following: D(5,2) = 0.1 , D((p5,p2),p3) = 0.3, D(1,4) = 0.5, D((p1,p4),((p5,p2),p3),p1) = 0.615.

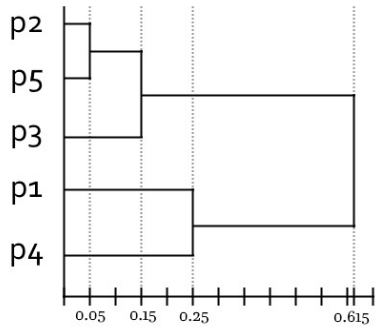

Figure 2: Dendrogram obtained using the UPGMA method

2 Introduction

Descriptive Analysis

The first step to take when handling big datasets is to pre-process them. Because of the computational complexity of the clustering methods,we sample our datasets, so that we can have a properly formed, smaller, workable subset. We applied random sampling. This method was chosen after verifying that the mean and variance of the sampled features are not significantly different from these of the original, complete dataset. It should be noted that this was also the case with the 3d features, which do not follow a simple normal distribution. The random choice of observations was done using pseudo-random numbers. In order to maintain consistency among the experiments and ensure quality sampling, specific seeds were used for the generation of the pseudo-random numbers that are used in the sampling. Scaling was also used to counter the issue of the features' magnitude differences, so as to make clustering possible. Refer to the Appendix for detailed plots of the datasets' and the samples' distribution and tables of their descriptive statistics.

2.1 Difficulties

First and most important difficulty in applying clustering methods to these two representations of the dataset, is the size of the dataset, specially because the space and time complexity of some of the algorithms like the Hopkins statistic or the hierarchical clustering methods are made for smaller datasets since some of them are in the order of $O(n^3)$, so some of them would take us quite a long time to process or we simply do not have enough memory to process them. In more detail, the features in the 6d dataset were all clearly following a normal distribution. The 3d features on the other hand did not have such a close approximation, as multimodality and significant skewness were observed. Another issue was the significant difference in the order of magnitude between the features, which invalidates almost any attempt to properly cluster our data. In order to overcome these issues, we used sampling and scaling of the data.

2.2 Differences between the two data representations and the Hopkins Statistic

We then moved on to apply PCA on both representations so as to form a first impression of our datasets' clustering tendency and detect any important differences. Reducing both 3d and 6d representations to 2d, allowed us to plot the datasets and provided us with a visual estimation regarding the results of the upcoming clustering attempts. More specifically, the 3d representation of the dataset is estimated to exhibit high clustering tendency, contrary to the 6d representation. See figures 3 and 4 below.

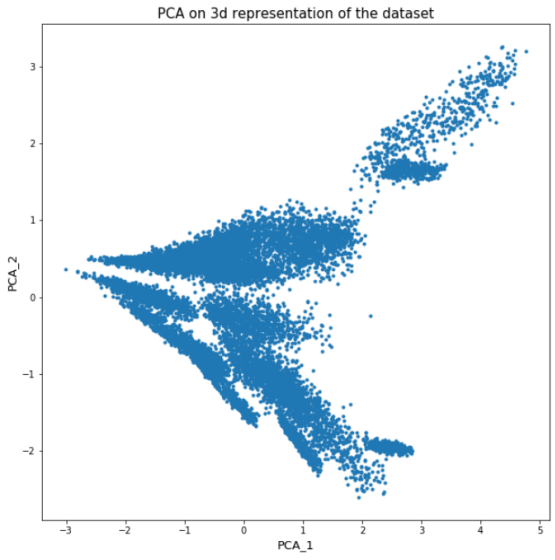

Figure 3: Scatter plot after applying PCA to the 3d Dataset

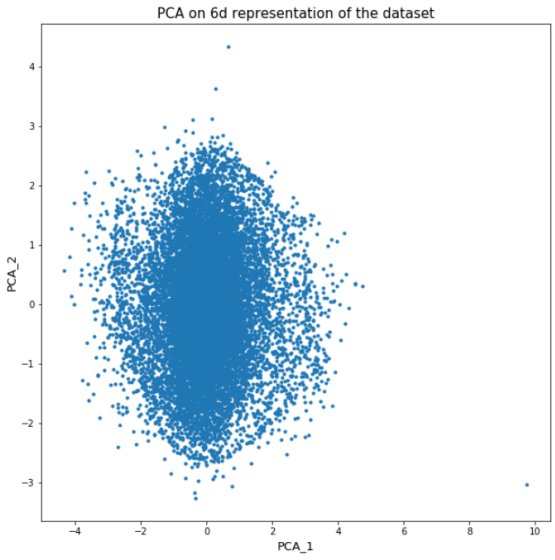

Figure 4: Scatter plot after applying PCA to the 6d Dataset

These estimations were further supported by the Hopkins statistic. Since the Hopkins statistic is a computationally demanding algorithm and our dataset has 300,000 records, we are going to need to sample data from the data-set to get a clear representation of the value of this statistic. We will use one of the most powerful techniques in sampling **Bootstraping**. This will allow us to construct the sample distribution of the Hopkins statistic. For this we are going to take a small sample of our data and repeat random sampling with replacement 150 times of a tiny data sampled

dataset. In each of these iterations we will construct the Hopkins statistic using our own implementation in python and then save the result on a vector. Afterwards we will plot the histogram of the results (fig. 8 and 9 in the appendix). On a side-note, this is possible because of the central-limit theorem that states that the sampling distribution of a parameter of a population tends towards a Gaussian distribution as the number of samples increases. As we can see the value of the Hopkins Statistic follows a somehow Gaussian distribution and leans towards the value **0.896**. Since the mean of the Hopkins Statistic is close to 0.9, we can consider this dataset to be highly "clusterable". We will do the same for the 6d dataset.
This time the normal behaviour of the sampling distribution is much more clear, the true value of the Hopkins Statistic tends towards **0.7** so we could consider this data to be borderline "clustreable". Another important note is that the Hopkins Statistic compares the euclidean distances of our data to a n-dimensional hypercube, it measures how close to a uniform distribution our data is. One example of a high Hopkins statistic on a non-clusterable dataset is to apply this technique to data from a Gaussian distribution. It will not be uniform since is Gaussian, but it is not highly "clusterable". That's why we must take this approach with caution.

2.3 Clustering dendrograms of different linkage methods

To have a better insight of the number of clusters we should consider we can perform a hierarchical clustering technique since this method does not need assumptions about the number of initial clusters to consider. The trade-back is that the space-complexity of the algorithm is $O(n^3)$ so running this on our laptops is not viable. Instead we are going to try to get a representative sample of our data of X data points to be able to run this method.
The Hierarchical clustering algorithm's results depend on what linkage strategy we determine to be the best one. We performed **single**, **average**, **complete** and **centroid** and **ward** linkage strategies. We observed that the single linkage method tends to cluster all of our points into one huge cluster 20, on the other hand complete linkage gave us a nice representation 21 on how clustering could work on the the whole dataset, complete linkage might be the best strategy because it usually yields clusters that are well separated and compact and since we are looking for galaxies we need well separated that represent each one a galaxy. Here we display the single and complete linkage, of both the 3d and the 6d representation of the dataset, which present the most differences and thus the greater study interest. (Refer to the appendix, figs. 10 - 13, for a detailed view of the dendrograms)

2.4 Determining best linkage strategy

Given the structure of the data, both in 3d and 6d representations, we expect for some of the linkage methods to be far more effective than others. Additionally the inherent disadvantages of some methods become more apparent, due that structure.
More specifically for the 6d representation, the scatter plot in fig.4 shows the data tend to cluster in one big cluster. After applying single linkage we see that the tendency of single link to form elongated clusters makes it a highly inappropriate method for the given dataset. As seen in fig. 12 clusters that are close to each other are continuously included in one cluster. Only after the shown cut point can we detect a slightly more significant distance between different clusters but at that point there are only five clusters remaining, which means that the linkage method has already united all points under a very small number of clusters. The same effect can be observed in the 3d data. Even though their structure displays better clustering tendency as seen in fig.3, elongated clusters are created again, by combining clusters whose distance measure does not differ greatly, as we can see in fig. 10. The first merges above the cut point are very close together, compared to the last merges. Conclusively, we can say that single linkage leads to clustering small numbers of clusters, with significant distance differences only when the last clusters are merged, which is to be expected as the whole dataset is combined in just a few clusters.

Complete linkage on the other hand allows for more elaborate clustering. Due to the distance metric of this method, the data are being clustered in many different clusters which will not be merged right away into one big cluster. Each cluster will incorporate data points that are on its close proximity and, contrary to simple linkage where one cluster would continuously incorporate single data points or small clusters, leading to elongated shapes. That becomes clear when we compare the figs. 10 and 12 with the figs. 11 and 13 respectively. Even at the last stages of the clustering procedure, several different clusters are maintained, with considerable distance measure between them.
Based on these two methods we can infer that the complete linkage is superior to the single linkage as it allows for different structures to be formed and smaller clusters are not as easily incorporated into bigger ones, allowing for a more representative structure of the data to emerge. (for an overview of the dendrograms of the other methods refer to the appendix, figs 14-19)

3 K-Means Clustering

3.1 Difference between 3D and 6D

The 3D data is more spread out around its three dimensions, and in opposition the 6D data appears more centered around a line with a few values spread out around this line. Additionally, the six dimensional data increases the runtime of the kmeans algorithm, given that the complexity of kmeans clustering greatly increases with higher k and dimensionality.

3.2 Suspected number of clusters

The sum of squares as well as the silhouette are used as performance measures for the unsupervised clustering. The sum of squares (SSE) tends downwards by the number of clusters (fig. 5), generally implying better clusters. However, the silhouette score should increase for better clusters which, seen from figure 6, does not happen. Given the relatively steady decrease in SEE, we limit our search between 25 and 40 clusters where the SSE starts stagnating, if anywhere. This attempts to get a good model with a low number of clusters (and their associated computational and statistical complexity).Within this region, the silhouette score is highest for 28 clusters, which is adopted to be the desired number of clusters.

3.3 Other distance measures

Choosing the right metric is crucial when we measure the similarity between our data points. Euclidean distance is simply the distance obtained when we use a ruler to measure the distance between two points. This metric can be calculated by using Pythagorean formula 2. Manhattan metric takes the absolute differences of the Cartesian coordinates, formula 3. Below are the basic formulas for 2D for each distance metric measure

$$\text{Euc}(X,Y) = \sqrt{\sum_{k=1}^{m}(X_{ik} - Y_{jk})^2} \tag{2}$$

$$\text{Man}(X,Y) = |X_{ik} - Y_{jk}| \tag{3}$$

Where ith data point belongs to kth cluster and m is the number of dimensions. We used R implementation of k-means algorithm to compare Euclidean and Manhattan distance metrics. Overall the convergence was acquired with higher iterations for the Euclidean distance metric compared to the Manhattan distance metric. This is due to the fact that the more sparse the data is, the Euclidean distance will exaggerate the already existing discrepancy on the dimensions with larger range of values, while Manhattan distance metric will minimize this discrepancy. In this case, considering computational complexity which is directly proportional to the number of iterations, calculating distances with Euclidean metric takes more time to deal with dimensions with larger range of values compared to the Manhattan distance metric.

4 Gaussian Mixture Models

4.1 Pseudo-code

For the pseudo code on Expectation Maximisation for Gaussian Mixture Models (GMM), we follow the solutions derived in [1]. The GMM assumes that the data is generated from a mixture of M gaussians as represented using:

$$p(x|\Theta) = \sum_{i=1}^{M} \alpha_i p_i(x|\theta_i) \tag{4}$$

. Here, the parameters θ_i of each model are the mean and covariance of each Gaussian component. Moreover, α_i is the weight given to the i'th Gaussian density function specified by $p(x|\theta_i)$, predicting all of the data. In said paper,

the updates for the parameters of the GMM model is calculated to be:

$$\alpha_l^{new} = \frac{1}{N}\sum_{i=1}^{N} p(l|x_i, \Theta^g) \tag{5}$$

$$\mu_l^{new} = \frac{\sum_{i=1}^{N} x_i p(l|x_i, \Theta^g)}{\sum_{i=1}^{N} p(l|x_i, \Theta^g)} \tag{6}$$

$$\Sigma_l^{new} = \frac{\sum_{i=1}^{N} p(l|x_i, \Theta^g)(x_i - \mu_l^{new})(x_i - \mu_l^{new})^T}{\sum_{i=1}^{N} p(l|x_i, \Theta^g)} \tag{7}$$

. In equations 5 to 7, the terms $p(l|x_i, \Theta^g)$ is the marginal probability of the unobserved data l with $l \in \{1, .., M\}$ depending on data point x_i and the guess for the parameters Θ (Θ^g), that is, the mean and covariance for the gaussians. α_l^{new} is the new weight for the l'th gaussian in equation 4. Similarly, μ_l^{new} and Σ_l^{new} are the new mean and covariance for the l'th gaussian respectively. Crucially, $p(l|x_i, \Theta^g)$ needs to be computed. The computation of $p(l|x_i, \Theta^g)$ uses bayes rule to derive that:

$$p(l|x_i, \Theta^g) = \frac{\alpha_l^g p_l(x_i|\theta_l^g)}{p(x_i|\Theta^g)} = \frac{\alpha_l^g p_l(x_i|\theta_l^g)}{\sum_{k=1}^{M} \alpha_k^g p_k(x_i|\theta_k^g)} \tag{8}$$

as seen in [1]. Thus, equation 8 requires the current or initial guess for the weights α_l and the probability of x_i given θ_k^g or θ_l^g are readily calculated as $\mathcal{N}_{\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l}(X = x_i)$; that is the probability of x_i given a normal distribution with mean μ_l and covariance matrix Σ_l. Using the stated equations, the pseudocode is in listing 1 in the appendix.

4.2 Applying GMM

4.2.1 Initialisation

The means and covariance matrices are initialised by applying kmeans clustering before running the expectation maximisation, resulting in an initial clustering with a corresponding initial state.

4.2.2 Likelihood function evaluation

In order to calculate the likelihood function, we use the equality derived in [1] and shown in equation 9.

$$\mathcal{L}(\Theta|\mathcal{X}) = \prod_{i=1}^{N} p(x_i|\theta) \tag{9}$$

The implementation returns the log likelihood so that summation can be performed. The likelihood indicates how well the data is predicted given the mixture model. GMM models are trained using kmeans and random on both 3D and 6D data, using a fixed subset of 5% of the full data. The results are given in table 5. The table shows that indeed, initialisation using kmeans results in a better optimised model with larger likelihood, although no statistical test is used. Also, the likelihoods for the 6D dataset are much lower than those using 3D data.

Table 5: Performance by data likelihood given the Gaussian Mixture model. All models were initialised using 28 clusters and 10 samples are taken from a fixed 5% of data.

	mean	standard deviation
kmeans 3D	0.2885	2.725e-3
random 3D	0.2824	8.252e-3
kmeans 6D	6.318e-4	1.390e-5
random 6D	6.046e-4	1.674e-4

4.3 Cluster evaluation

The cluster assignments are given in the included file *Team_14_clustering.csv* which gives the clustering for the 3D data. For that data, the mode cluster of 15 clusterings with 28 clusters was chosen. We also include fig, 7 in the results, that shows a distance matrix of the cluster means. The matrix clearly shows clusters 13 and 20 far removed from nearly all other clusters. Moreover, clusters 23-25 lie in each others proximity. As diffuse clusters indicate early merges, we also list the cluster SSE in table 6. The table shows very large distance to the cluster mean for cluster 2 (zero-indexed). Other diffuse clusters are 0,1,4,5,7 and 17. These clusters with large SSE are likely indicating early mergers, although a comparison with known mergers is needed. Conversely, the most recent merges are mostly indicated by clusters 25 and then 21,23,27.

5 Results

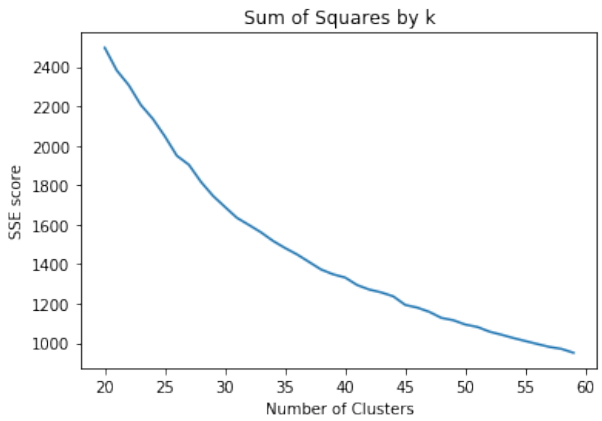

Figure 5: Sum of square distances to cluster means by number of clusters. Larger distances mean imply worse clusters.

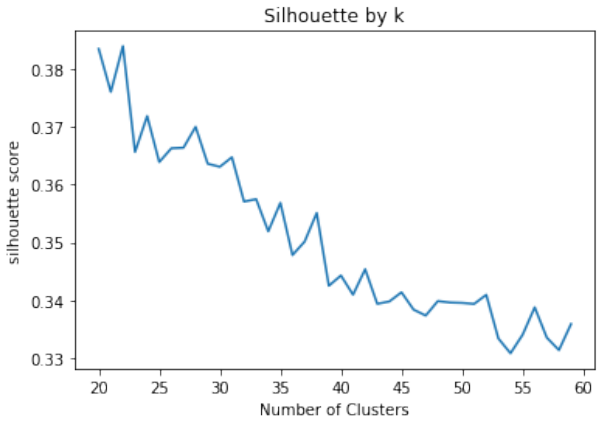

Figure 6: Silhouette score by number of clusters. The silhouette score is higher for better clusters so the performance goes down with more clusters

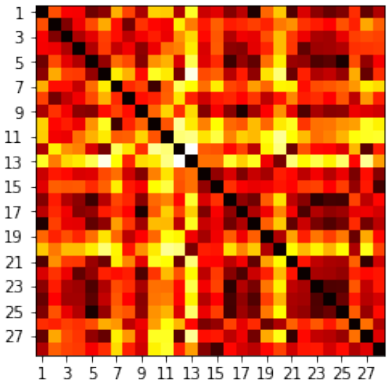

Figure 7: Distance matrix calculated from the cluster means. Brighter indexes indicate higher distances.

6 Discussion

With large datasets, subsampling and clustering methods allow working with them. As for hierarchical clustering, single linkage clustering was found inappropriate and instead, complete linkage turned out to be more representative. Using unsupervised evaluation on k-means clustering results in an performance curve over the number of clusters. From the performance by number clusters, we estimated there to be 28 clusters, although there is no strong evidence to such a claim. That is, 28 clusters was chosen as a trade off between cluster performance and model complexity. Using information derived from k-means clustering also helps when training a GMM using expectation maximisation, by initialising the Gaussians. By calculating the SSE for obtained clusters, estimates are given as to which clusters indicate early merges and which clusters belong to recently merged galaxies.

References

[1] Jeff Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical report, 1998.

7 Appendix

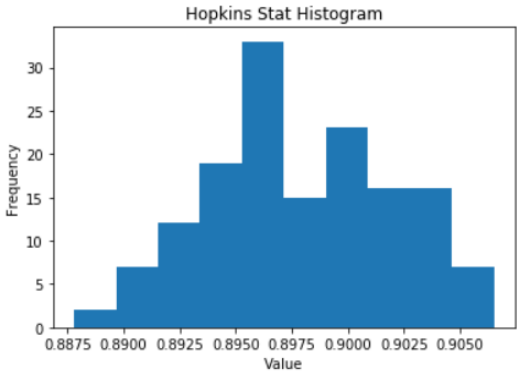

Figure 8: Hopkins Statistic histogram of the 3d Dataset

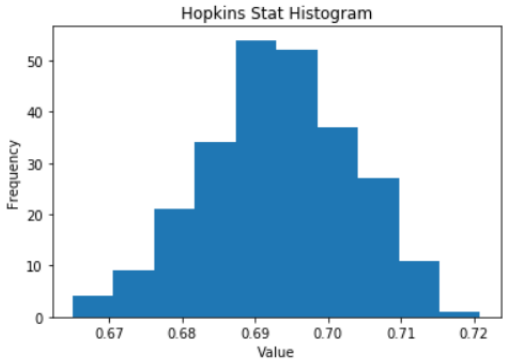

Figure 9: Hopkins Statistic histogram of the 6d Dataset

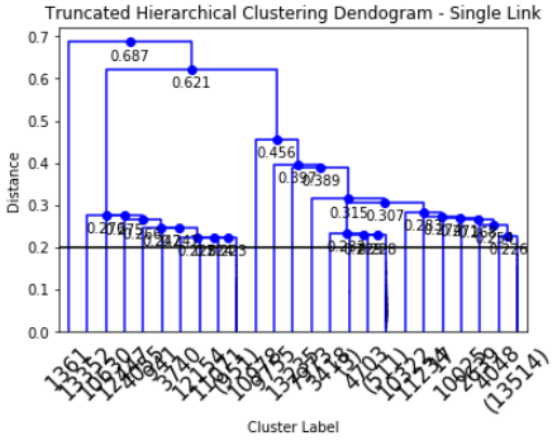

Figure 10: Single Linkage clustering results of the 3d Dataset

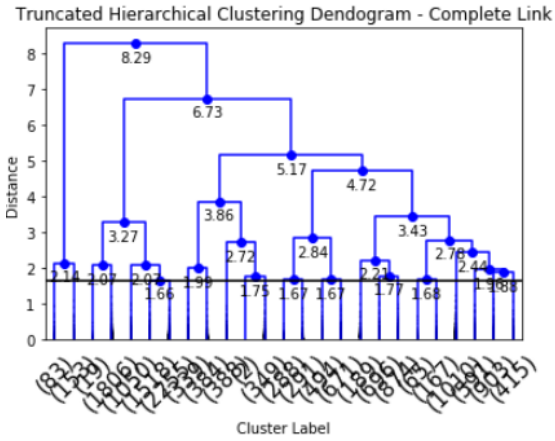

Figure 11: Complete Linkage clustering results of the 3d Dataset

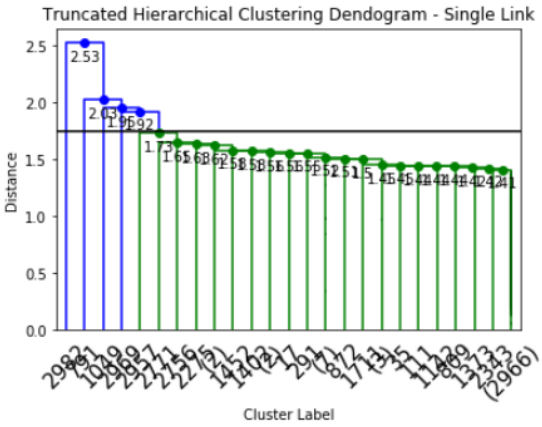

Figure 12: Single Linkage clustering results of the 6d Dataset

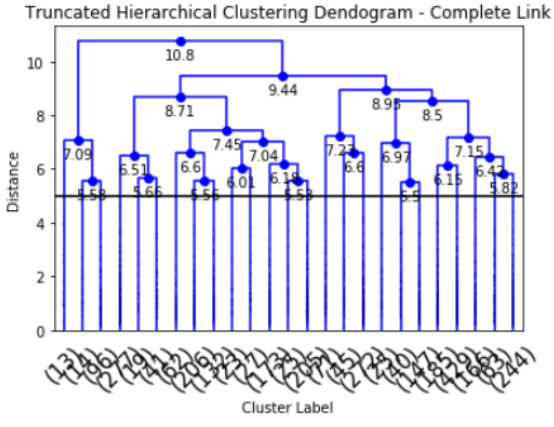

Figure 13: Complete Linkage clustering results of the 6d Dataset

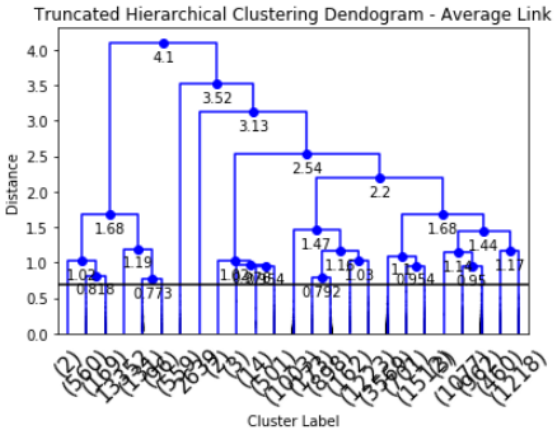

Figure 14: Average Linkage clustering results of the 3d Dataset

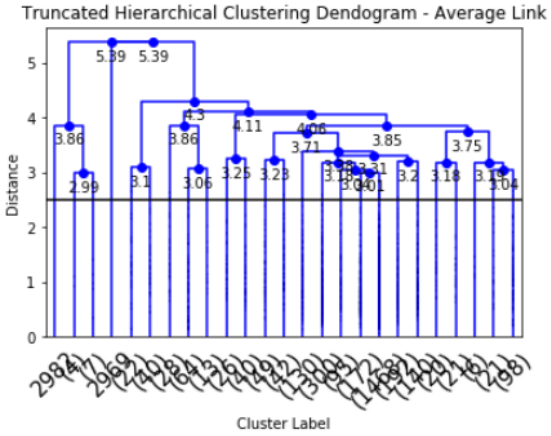

Figure 15: Average Linkage clustering results of the 6d Dataset

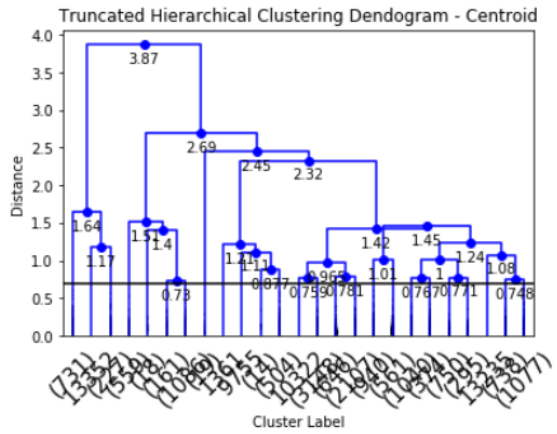

Figure 16: Centroid Linkage clustering results of the 3d Dataset

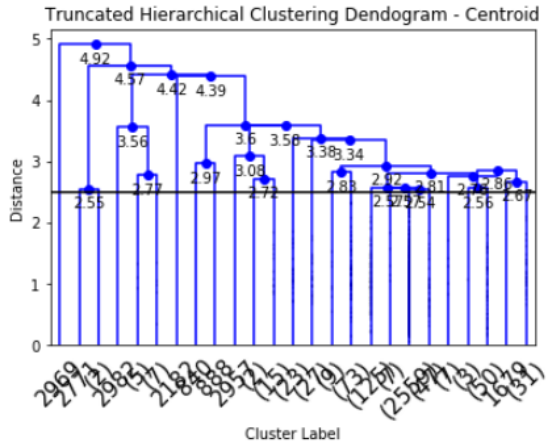

Figure 17: Centroid Linkage clustering results of the 6d Dataset

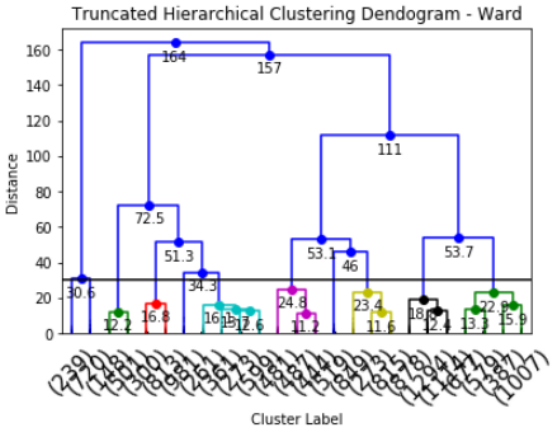

Figure 18: Ward Linkage clustering results of the 3d Dataset

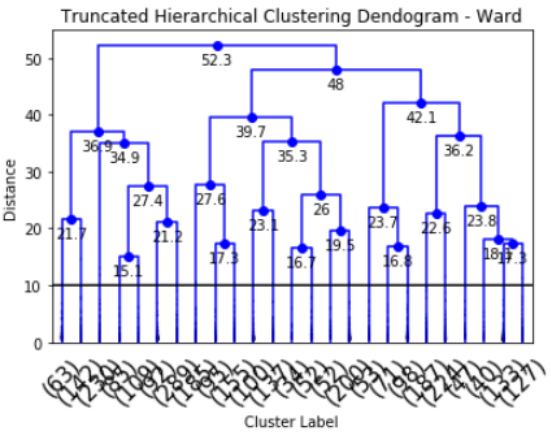

Figure 19: Ward Linkage clustering results of the 6d Dataset

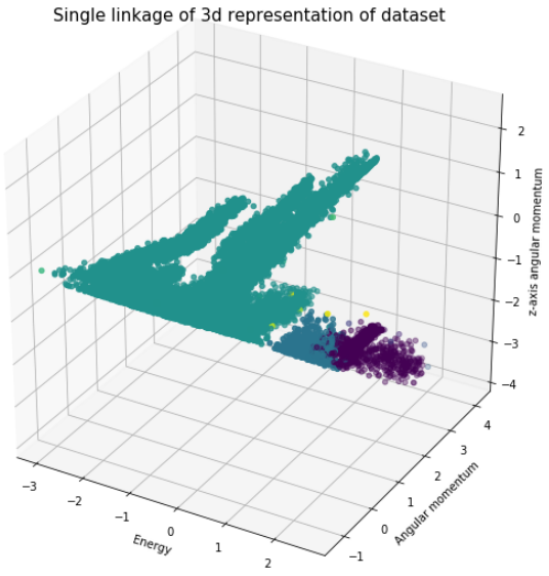

Figure 20: Single Linkage Clustering scatterplot of the sampled 3d Dataset

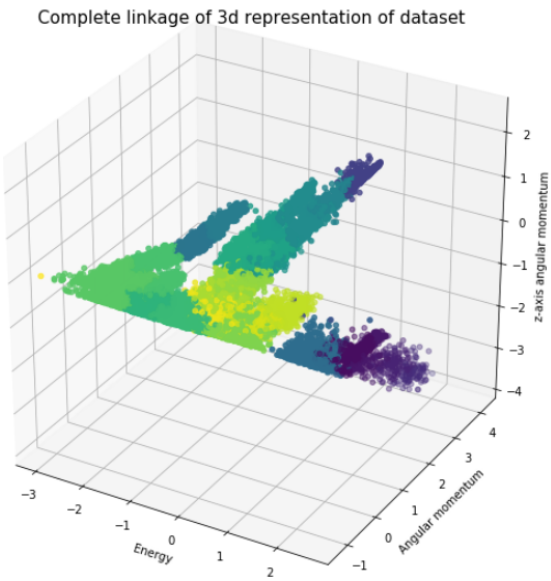

Figure 21: Complete Linkage Clutering scatterplot of the sampled 3d Dataset

Table 6: Sum of squared distance to cluster mean for each cluster, indicating early merges.

cluster	SSE
0	16842.0534
1	56790.9414
2	144286.4359
3	2116.9421
4	15606.1652
5	26285.5832
6	4625.1775
7	90874.9017
8	205.3687
9	3474.8766
10	74719.7095
11	4077.6521
12	4815.3569
13	1760.2501
14	1477.3588
15	6643.1343
16	1368.854
17	18006.4982
18	986.7723
19	720.726
20	214.3615
21	2.2775
22	406.0473
23	2.3244
24	102.5682
25	0.8805
26	27.8287
27	3.4797

```
import numpy as np
from scipy.stats import multivariate_normal as normal

# assume data is given as x (N X dimensionality)
# Specify number of gaussians
M               = 20
N               = len(x)
dimensionality  = 3

# Guess the initial parameters for the gaussians. This creates uniform random matrices.
# The gaussian means can be initially guessed according to other clustering approaches.
# The means should also be at least scaled to the range of data.
alpha   = np.array([random() for _ in range(M)])
mu      = np.array([np.random((dimensionality, )) for _ in range(M)])
sigma   = np.array([np.random((dimensionality, dimensionality)) for _ in range(M)])

# function returning p(l|x_i, theta^g) using the formulas presented in text
# normal.pdf returns the probability of data given the gaussian
def calc_pl(data, l):
    p_xi = np.sum([alpha[k] * normal.pdf(data, mu[k], sigma[k]) for k in range(M)])
    return alpha[l] * normal.pdf(data, mu[l], sigma[l]) / p_xi

# Keep track of differences per iteration for stopping criterion
diff = {'alpha': 9999, 'mu':9999, 'sigma':9999}

# Difference threshold topping criterion, to tune for given  data.
thresh = {'alpha':1, 'mu':1, 'sigma':1}

# Expectation and maximisation
while diff['alpha'] > thresh['alpha'] or diff['mu'] > thresh['mu'] or diff['sigma'] > thresh['sigma']:
    #reset differences to 0
    diff = {key : 0 for key in diff.keys()}

    for l in range(M):
        # sum of marginal probabilities for all datapoints
        marg_sum = np.sum([calc_pl(x[i], l) for i in range(N)])

        # Find new parameter values using given formulas
        alpha_new[l] = 1 / N * marg_sum
        mu_new[l] = np.sum([x[i] * calc_pl(x[i], l) for i in range(N)], axis=0) / marg_sum

        sigma_new[l] = np.zeros((dimensionality, dimensionality))
        for i in range(N):
            diff = x[i] - mu_new[l]
            sigma_new[l] += calc_pl(x[i], l) * diff * diff.transpose()
        sigma_new[l] /= marg_sum

        # Calculate absolute differences
        # Summing over dimensions works better when dimensions are scaled
        diff['alpha'] += np.abs(new_alpha[l] - alpha[l])
        diff['mu'] += np.sum(np.abs(mu_new[l] - mu[l]))
        diff['sigma'] += np.sum(np.abs(sigma_new[l] - sigma[l]))

        alpha[l]   = alpha_new[l]
        mu[l]      = mu_new[l]
        sigma[l]   = sigma_new[l]
```

Listing 1: Python prototype for creating a Gaussian Mixture Model. Notes: np.sum calculcates the sum of an input matrix and over an axis when one is specified.