Clustering Algorithms: A Comparative Study

Soham Dutta[a,1], **Aishik Bhattacharya**[a,1], **Soham Chatterjee**[a,1], **Sauparna Kar**[a,1]

[a]*Indian Statistical Institute, Kolkata*

Professor Arnab Chakraborty

Abstract—Clustering Algorithms aim to group data into coherent groups based on the underlying patterns present in the data. Several clustering methods are known: each focusing on one aspect of the data and having its own merits and demerits. In this project, we apply several clustering algorithms to an astronomical dataset and present a comparative study among them.

Contents

1. Introduction

A galaxy cluster is a large collection of galaxies spanning over several billion light years. Galaxy clusters are some of the largest gravitationally bound structures in the observable universe and typically contain 100 to 1000 galaxies. These structures are extremely important for the study of dark matter and dark energy and subsequently cosmic inflation. Attempts to cluster galaxies have been made since long ([4], [3], [5]). The most recent attempt at galaxy clustering is based on an ML approach due to Qiu et. al ([6]). Here, we apply several clustering algorithms, namely K-Means, K-Median, DBSCAN and EM Algorithm (Gaussian Mixture Models) to an astronomical dataset. We propose a new clustering algorithm (RCICA) which is a modification of the K-Means Algorithm. The algorithms are compared based on the Davies-Bouldin Index [2] [7]. Furthermore, a novel method to estimate the the centre of masses of the galaxy clusters is given which helps in the study of mass distributions of the clusters.

2. Data

The data we worked on has been obtained from the Sloan Digital Sky Survey with the search parameters TOP 5000 and CLASS='GALAXY' and S.z>0. The distance R was obtained by

$$\mathrm{R} = \frac{\text{redshift} \times \text{speed of light}}{\text{Hubble's Constant}}$$

Taking RIGHT ASCENSION as $\frac{\pi}{2} - \theta$ (in radians) and DECLINATION as ϕ (in radians), the polar coordinates were obtained as

$$\mathrm{X} = \mathrm{R}\sin\theta\cos\phi, \mathrm{Y} = \mathrm{R}\sin\theta\sin\phi, \mathrm{Z} = \mathrm{R}\cos\theta.$$

3. Methodology

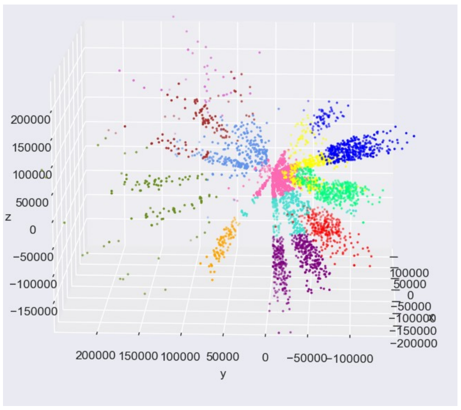
3.1. K-Means

The K-Means algorithm essentially aims to "cluster" the dataset into k clusters based on a suitably chosen metric. In this case, we work with the standard Euclidean Norm i.e. $d(\vec{x},\vec{y}) = \sqrt{(x_1-y_1)^2+(x_2-y_2)^2+(x_3-y_3)^2}$. We start with 12 suitably chosen points as the initial centers for the 12 clusters. The algorithm is essentially two-step which is listed as follows:

1. **(Assignment Step)** For each data point $\vec{i}$, compute $d(\vec{i}, \vec{c_j}^{(t)})$ where $\vec{c_j}^{(t)}$ denotes the cluster centers at the t-th step. If
$$\hat{j} = \arg\min_j d(\vec{i}, \vec{c_j}^{(t)}),$$
assign $\vec{i}$ to cluster number $\hat{j}$.
2. **(Updation Step)** For each cluster $J^{(t)}$ with center $\vec{c_j}^{(t)}$, redefine the centers $\vec{c_j}^{(t+1)}$ as
$$\vec{c_j}^{(t+1)} = \frac{\sum\limits_{\vec{x_k}\in J^{(t)}} \vec{x_k}}{|J^{(t)}|}$$

In case of our dataset, the K-Means algorithm converges pretty quickly i.e. in about 35-40 iterations.

The results obtained are shown below:



The initial values and the final obtained values of the centers of the clusters are given below:
INITIAL VALUES : [[-148188/2, -116932/2, 125808/2], [-179866/2, -133570/2, 16842/2], [-189366/2, -123086/2, -99026/2], [-138994/2, -67838/2, -165420/2], [-106588/2, -25146/2, -220526/2], [-92576/2, -95192/2, 165162/2], [-174980/2, 67246/2, -108338/2], [-89138/2, 153372/2, -49446/2], [-104482/2, 174496/2, 20206/2], [-79914/2, 138470/2, 94828/2], [-78114/2, 126436/2, 177872/2], [-28674/2, 50522/2, 206538/2]]
OBTAINED VALUES : [[-99558.24379432251, -93816.9417305628, 92471.00725097724],[-139638.1602094458, -76421.12844215108, 24420.93814725363],[-141141.01277674694, -68340.22293990494, -49193.35727220487], [-55021.89437775478, -22252.747411751574,

-50402.59687961611],[-93167.28236907773, -28804.136073316768, -129764.28532534826],[-58060.779088486895, -48908.37998737092, 34723.67462545727],[-130128.44883057548, 54205.711512838505, -82596.92903000841],[-18382.551592347263, -11321.146877634732, -1713.0423641561251],[-51618.033340700444, 142175.30326360065, -36366.159941158425], [17396.85710297639, 39949.73316516686, 29622.347291205995], [-47591.31799446712, 86163.33026459278, 112604.91970473301], [75458.48984021915, 94747.85190701825, 159561.28271223168]]

3.2. K-Median

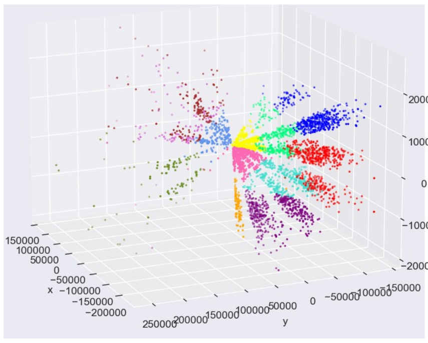
As the name suggests, this is a very close relative of the K-Means Algorithm. The algorithm is almost the same except for the Updation Step wherein we use

$$\vec{c_j^{(t+1)}} = \text{median}\ (J^{(t)})$$

This is a really small but subtle change. Since median is a more robust estimator of central tendency than the mean, the K-Median Algorithm is more "immune" to outliers than the K-Means Algorithm.

In this case too, the algorithm converges in about 23-25 iterations.

The results are shown below:

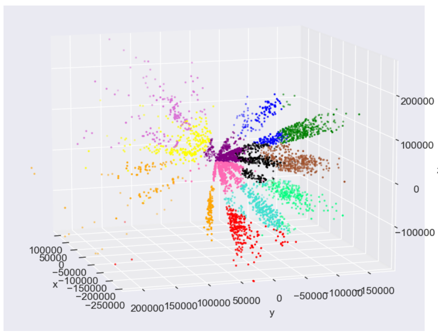


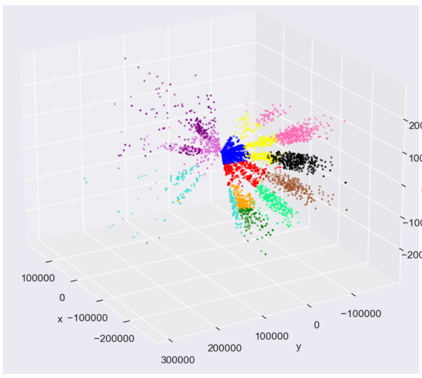
3.3. RAndomized CEntroid Algorithm (RACE)

This is a spin-off to the K-Means Algorithm. The algorithm is the same except, this time, for the Assignment Step. In the K-Means Algorithm, we explicitly specified the $\vec{c_j^{(1)}}$ vectors. However, in this algorithm, we generate 12 points at random from [1, 5000] and set those to be our initial cluster centers. Since the RANDOM.RANDINT command in Python follows the Discrete Uniform Distribution, we are essentially picking up 12 points $\{\vec{c_j^{(1)}}\}_{j\in\{1,2\cdots,12\}} \sim \mathcal{U}\{0,\cdots,4999\}$.

Since the initial cluster centers vary with each run of the code, we don't get the same clusters everytime we run the code. It might happen (with a very very low probability) that one of the intial cluster centers is an outlier which then renders that point itself to be a single cluster.

A few examples of the outputs obtained using the above algorithm is shown below:

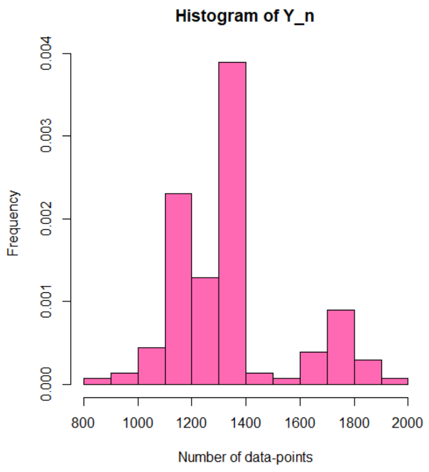




How so ever random this might look, there is something interesting lurking beneath. Define a sequence of random variables $\{Y_n\}_{\mathbb{N}}$ as follows:

$$Y_n = \#\text{ points present in the cluster containing the maximum number of data-points in the } n\text{th iteration of RACE}$$

The algorithm was run for around 500 times and a histogram of $\{Y_n\}_{\mathbb{N}}$ was plotted. This histogram looks like:



The number of points in the cluster with the maximum of number of points found out through the K-Means Algorithm is roughly around 1300. We see that here also the histogram has a pretty prominent peak around 1350. Furthermore, there is a small peak at around 1800. One probable explanation of this is that while randomly choosing the centers, in some cases the cluster with the maximum number of points is a conglomeration of two smaller clusters. This might cause the number of datapoints to shoot up to 1800.

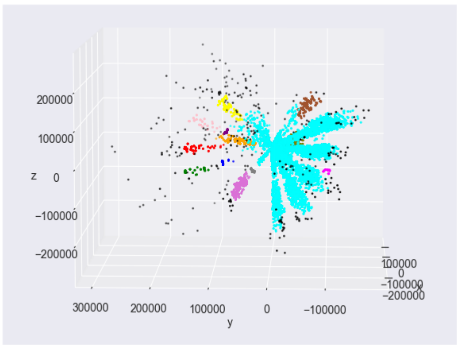
3.4. DBSCAN

Density-Based Spatial Clustering of Applications with Noise is a data clustering method that is commonly used in machine learning and data mining applications. Unlike K-Means and K-Median, it does not need a pre-defined number of clusters. It relies on two primary parameters, Epsilon (**eps**) which specifies the radius around each point to search for neighboring points and **MinPts**, the minimum number of points within the **eps** neighborhood of a point for it to be considered a **Core Point**.
Depending upon the **eps** and **MinPts** values, the DBSCAN algorithm classifies every data point into three categories:

1. **Core Point**:- A data point is a core point if there are at least **MinPts** number of points within the specified **eps**.
2. **Border Point**:- A data point is a border point if it can be reached from a core point and it has less than **MinPts** number of points in its **eps** neighbourhood.

3. **Noise Point**:- A data point which is not a core point itself and cannot be reached from any other core point is called a noise/outlier point.

Here, we have chosen **eps = 12000** and **MinPts = 5**. The resulting output is shown below:



There is no universal rule for choosing the optimal values for **eps** and **MinPts** as they depend on the characteristics of the dataset. Thus, we used the density and distribution of data points to determine the parameter values.

3.5. EM Algorithm (Gaussian Mixture Models)

The expectation maximization algorithm (EM algorithm) was proposed by Dempster et al. (1977) [1] and is a general iterative scheme for learning parameters (maximum likelihood or MAP) in mixture models and, more generally, latent-variable models.
In our example of the Gaussian mixture model, we choose initial values for $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k$, where they represent **mean vectors**, **covariance matrices** and **mixture weights** respectively, and keep updating until convergence between

1. **(E-Step)** Evaluate the responsibilities r_{nk} (posterior probability of data point n belonging to mixture component k).
2. **(M-Step)** Use the updated responsibilities to re-estimate the parameters μ_k, Σ_k, π_k.

For convergence, we can check the log-likelihood or the parameters directly. A concrete instantiation of the EM algorithm for estimating the parameters of a GMM is as follows:
1. Initialize μ_k, Σ_k, π_k.
2. E-Step: Evaluate responsibilities r_{nk} for every data point $\boldsymbol{x}_n$ using current parameters $\pi k, \mu_k, \Sigma_k$:

$$r_{nk} = \frac{\pi_k \mathcal{N}\left(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\sum_j \pi_j \mathcal{N}\left(\boldsymbol{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)}$$

3. M-Step: Re-estimate parameters π_k, μ_k, Σ_k using the current responsibilities r_{nk} (from E-step):

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \boldsymbol{x}_n$$

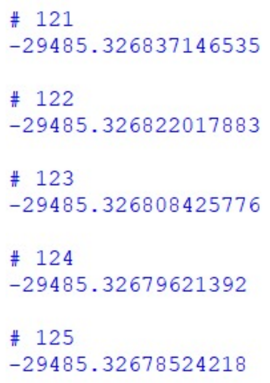
$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \left(\boldsymbol{x}_n - \boldsymbol{\mu}_k\right) \left(\boldsymbol{x}_n - \boldsymbol{\mu}_k\right)^{\top}$$

$$\pi_k = \frac{N_k}{N}$$

As with almost all EM problems in Statistics, the initialization of μ_k and Σ_k was done from the K-Means code. Our code requires almost 125 iterations to converge to around four digits after the decimal point. The high number of iterations required is probably due to the large number of parameters (36) we are estimating.

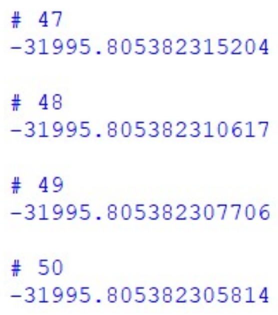
In order to make computation less costly, we can choose not to update the covariance matrices, thus reducing the number of parameters to be estimated to 24. In such a case, the EM Algorithm converges upto the 8th decimal place in under 50 iterations.

The observed log-likelihood (when the covariance matrices are also updated) of the last few iterations are listed below :



```
# 121
-29485.326837146535

# 122
-29485.326822017883

# 123
-29485.326808425776

# 124
-29485.32679621392

# 125
-29485.32678524218
```

When covariance matrices are not updated, the convergence is as follows:



```
# 47
-31995.805382315204

# 48
-31995.805382310617

# 49
-31995.805382307706

# 50
-31995.805382305814
```

4. Comparing the Different Algorithms

The Davies-Bouldin Index (DBI) [2] is a metric to evaluate clustering algorithms. The calculation of DBI is a three-step process which is listed below:

1. The cluster radius (r_i) is defined as

$$r_i = \sqrt{\frac{1}{|C_i|} \sum_{\vec{x} \in C_i} d(\vec{x}, \vec{c}_i)^2}$$

2. The quality of each cluster (R_i) is the quality of the worst pair i.e.

$$R_i = \max_{j \neq i} \frac{r_i + r_j}{d(\vec{c}_i, \vec{c}_j)}$$

3. The Davies-Bouldin Index is the mean of R_i's i.e.

$$\text{DBI} = \frac{\sum_{i=1}^{k} R_i}{k}$$

The DBI value is between 0 and 1. A smaller value of DBI indicates better separation of the clusters and "tightness" inside the clusters.

In case of the K-Means algorithm, the DBI turns out to be 0.07251012310726687 while in case of K-Median algorithm, the DBI turns out to be 0.1422064942107469.

Since the EM Algorithm is a probabilistic method i.e. it does not explicitly say what points each clusters should comprise of, we cannot calculate DBI in this case. Also, in case of DBScan, since it is a density-based clustering algorithm and thus also does not specify the exact centers of the clusters obtained, we are unable to calculate DBI in this case too.

5. Calculation of Center of Mass of the Galaxy Clusters

In this section, we propose a new method to compute the centre of mass of these galaxy clusters and provide applications and implications of such calculations.

5.1. Calculation of CoM

We know galaxies are majorly disc shaped. Each galaxy has its own **self energy** (energy required during the formation of the galaxy). For any radiating body, the radiation pressure must match the the gravitational pressure at each radius within the body. Hence, for these galaxies, **radiation power must be equal to their self energies**.

$$S_E \propto \frac{M^2}{R}$$

where S_E, M and R are the self energy, mass and radius of a galaxy. Unfortunately we do not have the data for the radius of the galaxies hence we take it to be equal for all of them.
So, $S_E \propto M^2$. Hence the radiation power, $P_R \propto M^2$.
Let the galaxy be at a distance d from the Earth. So intensity of radiation received at Earth,

$$I_E \propto \frac{M^2}{4\pi d^2} \implies \boxed{\log I_E \propto \log\left(\frac{M}{d}\right)}$$

Now, we have the data for log(intensity) taken through 5 different wavelength filters: ultraviolet (u), green (g), red (r), infrared (i) and near-infrared (z). Now, it might be cumbersome to work with 5 different intensity values. Hence we try to create an **Intensity Score** (I Score) to represent the intensity of light received here on Earth.

The greater the frequency of light radiated, the greater is its energy. That is radiation power of shorter wavelengths is greater than that of longer wavelengths. Since radiation power balances self energy, we can conclude, the higher the intensity in shorter wavelength filters, the higher will be the mass of the galaxy. Hence while calculating the Intensity Score, we apply priority weights to these intensities and take their mean. Hence

$$\log \text{I Score} = \frac{5\log I_u + 4\log I_g + 3\log I_r + 2\log I_i + \log I_z}{15}$$

Here, we run into a slight issue. The light obtained from each of the galaxies is red-shifted. Hence we need to find the true wavelength in order to assign proper priority weights to the log(Intensity) values. Since the red-shifted velocities are not available in our data, we can assume the median wavelength of each of the spectral regions to be the red-shifted wavelength, say λ_2. We need to find original wavelength, λ_1. We know,

$$\frac{\lambda_2 - \lambda_1}{\lambda_1} = r \text{ ,where } r \text{ is the red-shift}$$

$$\implies \lambda_1 = \frac{\lambda_2}{1+r}$$

We then check in which region of the spectrum λ_1 lies, and apply the appropriate priority weight. Now we substitute the value of log(I Score) in place of $\log I_E$ to calculate relative masses of the galaxies.

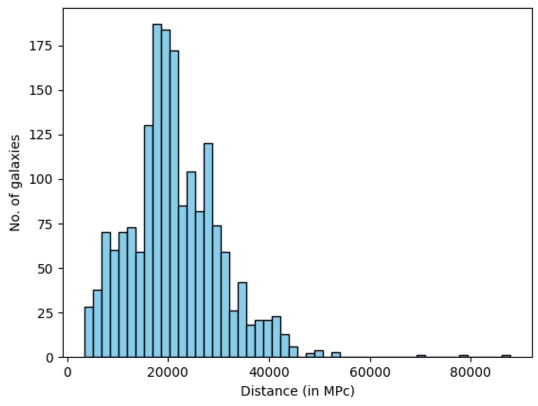
Note, to calculate the centre of mass of a cluster, calculating the relative masses of the galaxies is sufficient. Hence we assign the first galaxy in our dataset the mass of 1 unit, and calculate the masses of the other ones. So,we apply the **K-Means** algorithm to hard-cluster the galaxies. Thus, for each cluster, centre of mass is just

$$\overrightarrow{CoM} = \frac{\sum M_i \vec{r_i}}{\sum M_i}$$

where $\vec{r_i}$ represents the position vector of each galaxy in the cluster.

5.2. Distribution of Galaxies around CoM

Once we calculate the center of masses of each cluster, we compute the distance of each point present in a cluster from its corresponding center of mass. A histogram was plotted which is shown below:



6. Conclusion

In this project, we have tried to implement various clustering algorithms on an astronomical dataset. We have worked with both soft and hard clustering algorithms (probabilistic clustering and absolute clustering respectively). Furthermore, we have tried to randomize the intialization of the K-Means Algorithm. A novel method to calculate the relative masses of each galaxy and, in turn, the center of masses of each galaxy cluster. At last, an attempt to explore the distribution of distances of each galaxy from the center of masses of each cluster has been made. Our findings and methodology have been extensively presented in this report.

7. Limitations and Further Work

Since we don't have the data corresponding to the rotation and radius of the galaxies and also the exact wavelength of light observed, we have only been able to calculate the relative mass of the galaxies and not the absolute mass. If we had the aforesaid data, we could've calculated the absolute mass of each galaxy with high precision. Moreover, we are not really sure about the distribution of the distances of each galaxy present in a cluster from its corresponding center of mass.

Looking forward, it would be interesting to obtain the histogram of $\{Y_n\}_{\mathbb{N}}$ if the initial centers were chosen from some specified distribution instead of Discrete Uniform. Also, after calculating the center of masses of each cluster, one can consider each cluster to be a point mass which simplifies calculations related to Newtonian Mechanics. There is another considerable application of this calculations: if we consider the galaxy clusters near the edge of our observable universe, by treating them as point sources, we can calculate the effect of the forces originating from outside the observable universe and if we are able to find the distribution of distances of each point from the center of mass, we will be able to predict the structure of the "un"observable universe even though we cannot see it directly.

8. Acknowledgement

We would like to extend our sincere appreciation to all the individuals who contributed to the successful completion of this project. Each member of the team played a vital role, and their dedication, expertise, and collaborative spirit were invaluable throughout the entire process. We are also grateful to Prof. Arnab Chakraborty for their guidance, support, and invaluable insights, which were instrumental in shaping the direction of the project and ensuring its success. Additionally, we would like to acknowledge the support and understanding of our families and friends, whose encouragement and patience sustained us during the challenging moments of this endeavor. This project would not have been possible without the collective effort and dedication of everyone involved.

9. Codes and Outputs

All codes and outputs pertaining to our project can be found here.

References

[1] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm", *Journal of the royal statistical society: series B (methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[2] D. L. Davies and D. W. Bouldin, "A cluster separation measure", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979. DOI: 10.1109/TPAMI.1979.4766909.

[3] R. G. Carlberg, H. Yee, and E. Ellingson, "The average mass and light profiles of galaxy clusters", *The Astrophysical Journal*, vol. 478, no. 2, p. 462, 1997.

[4] J. A. Peacock, "The evolution of galaxy clustering", *Monthly Notices of the Royal Astronomical Society*, vol. 284, no. 4, pp. 885–898, Feb. 1997, ISSN: 1365-2966. DOI: 10.1093/mnras/284.4.885. [Online]. Available: http://dx.doi.org/10.1093/mnras/284.4.885.

[5] A. V. Kravtsov and S. Borgani, "Formation of galaxy clusters", *Annual Review of Astronomy and Astrophysics*, vol. 50, pp. 353–409, 2012.

[6] L. Qiu, N. R. Napolitano, S. Borgani, *et al.*, *Cosmology with galaxy cluster properties using machine learning*, 2023. arXiv: `2304.09142 [astro-ph.CO]`.

[7] H. Yin, A. Aryani, S. Petrie, A. Nambissan, A. Astudillo, and S. Cao, *A rapid review of clustering algorithms*, 2024. arXiv: `2401.07389 [cs.LG]`.