

GEO5017 A1: Clustering

Pratyush Kumar (5359252) , Cynthia Cai (5625483)

1 Introduction

This assignment is designed for the students of GEO5017 to understand the working of the unsupervised clustering algorithms, namely, k-means clustering, hierarchical clustering, and DBSCAN clustering. Through this exercise we also learn to arrange different features for clustering algorithms to work on, and in the process we also brainstorm upon how a feature can be made and why we think it should be utilised in the algorithmic clustering.

This assignment also brings an opportunity to program as a team, therefore enabling us to use collaboration and version controlling platforms like github. Moreover, this assignment also helped us to understand the usage of Jupyter Notebooks and Pandas in data science based projects.

The github repository for this group can be found [here](#).

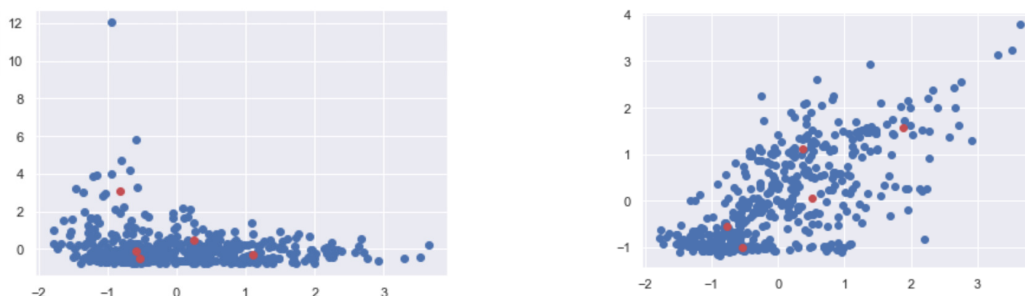
2 Methodology

2.1 Feature selection

2.1.1 Selection process

In various attempts, we extracted ten features, including variance in X/Y/Z values(3), range of X/Y/Z values(3), median height(1), volume of convex hull(1), 2D density of the object on x-y plane(1) and 3D volume density(1).

We used plot to judge which is more valuable. For instance, clustering through scipy can give us hints about feature selection.



(a) using 5 features, which are median of height, range of height, volume of convex hull, 2D footprint density, 3D volume density

(b) using 4 features, which are median of height, range of height, volume of convex hull, 3D volume density

Figure 2.1: *Different feature selection for K-means clustering plot via scipy*

Then we plotted against one another to briefly analyse if any visible characteristics can be distinguished between the features. From the plot, we can see that some of the features are already distinguishable (see the histograms plotted for each variable).

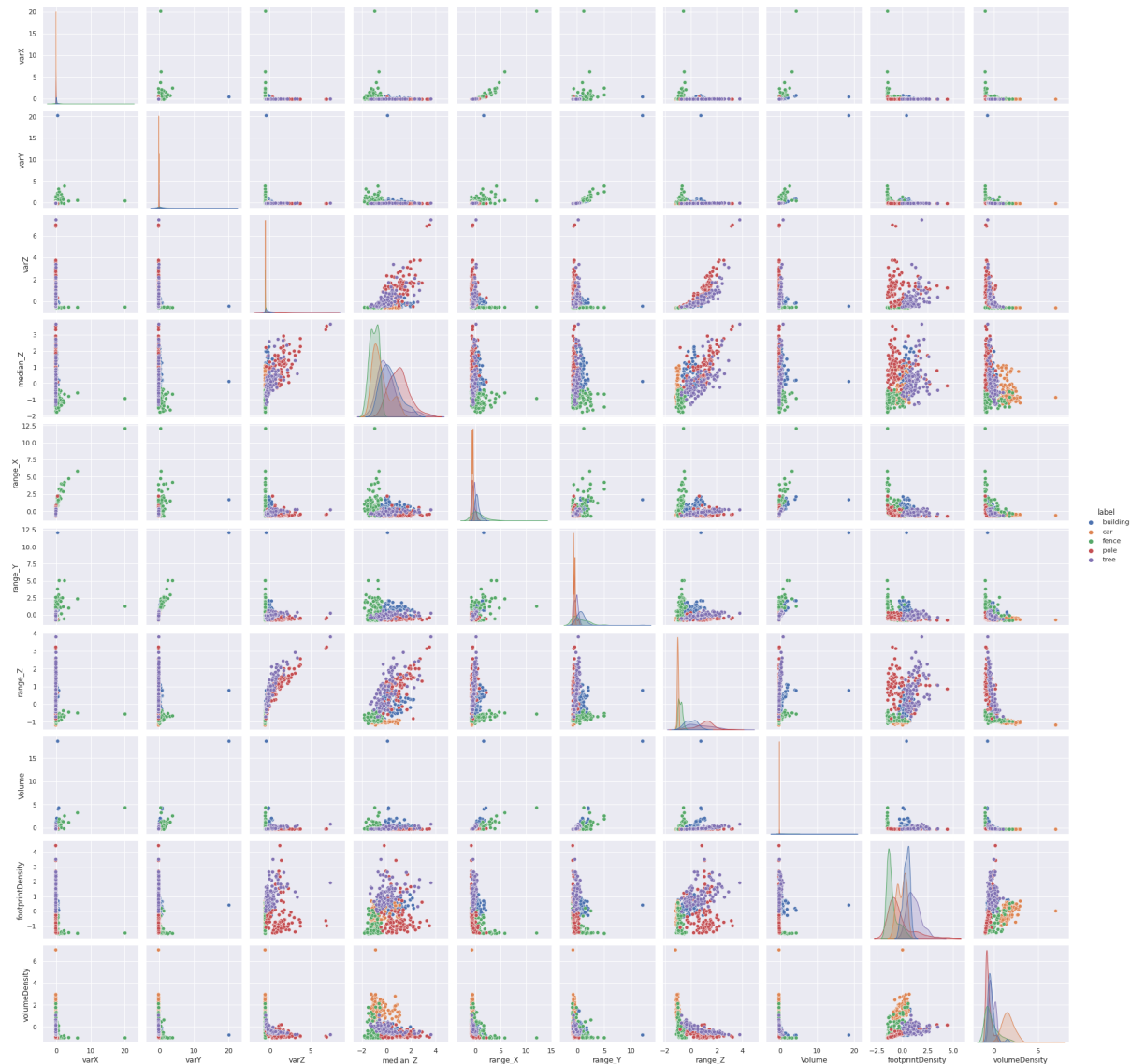


Figure 2.2: Graph of initial feature selections

2.1.2 Selection result

We eventually selected only four features for the following reasons:

Feature	Reason for selection
Median of height	Since objects like trees have more density on the upper portion of the object, we thought it would be nice to take the median Z value into account, thus for trees the median Z would be closer to the canopy than trees, this would be lower for cars and fences.
Range of height	Range of x,y,z was taken into consideration since different objects have different shapes, and apart from the variance, the range also gives us a fair idea of the spread of the points in the object

volume of convex hull	The convex hull of the object will be similar for similar objects, like trees will be in a typical range, buildings would have higher volume and cars would have a smaller volume.
3D volume density	The 3D density of the object will vary depending on the size of the object and number of points it has, given that the LiDAR points are collected uniformly, the density will also prove to be an important metric for classification.

Table 2.1: *Features to be used in the algorithm*

Then, all features were standardised by the following formula:

Standardised data = (original dataset - mean(original dataset)) / standard deviation (original dataset)

2.2 Distance

The distance between two objects can be calculated on the basis of their features. There are different ways to calculate distances between variables, some of them being:

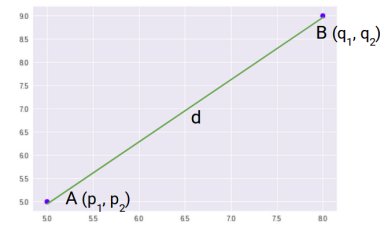
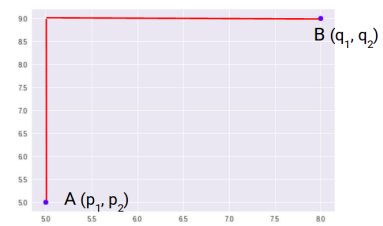
Name	Definition	Formula	Figure
Euclidean distance	the direct displacement between two vectors	$D_e = \left(\sum_{i=1}^n (p_i - q_i)^2 \right)^{1/2}$	
Manhattan distance	the distance between two points as if it were being calculated on the axes	$D_m = \sum_{i=1}^n p_i - q_i $	
Minkowski distance	a distance metric between the euclidean distance and the Manhattan distance. If the value of p is set to 1, it is essentially manhattan distance.	$D = \left(\sum_{i=1}^n p_i - q_i ^p \right)^{1/p}$	

Table 2.2: *The definition and types of distance*

Distances are calculated between these vectors in order to distinguish how similar two vectors are, based on their distance. In a matrix of distances between multiple features, we can distinguish which features are the closest and then cluster them.

2.3 Clustering Algorithm

2.3.1 K-Means [1]

Input:

Dataset $D = [x_1, x_2, x_3, \dots, x_m]$;
The number of clusters k .

Process:

Pick k elements from D as initial centroids [$centroid_1, centroid_2, \dots, centroid_k$]

Define `MaxInteraction` and `epsilon`

`Interaction = 0`

Repeat

`Interaction` plus one

 Generate kdtree of centroids

Let $C_i = []$ ($1 \leq i \leq k$)

For $j = 1, 2, \dots, m$ do

 Find the nearest centroid of x_j

 Assign x_j to that centroid's cluster

End for

For $i = 1, 2, 3, \dots, k$ do

 Compute new centroids by computing the mean of each feature

 If the distance of any new centroid and old one $> \epsilon$:

 Break

Until iteration $> \text{MaxInteraction}$

Output:

Clusters $C = [C_1, C_2, \dots, C_k]$

2.3.2 Hierarchical clustering

Hierarchical clustering functions by clustering pairwise objects or clusters together and builds the cluster till no object is left to be clustered. The following pseudocode elaborates the implementation of hierarchical clustering algorithm in our assignment.

```
// distanceMat is a pandas dataframe
distanceMat = distance matrix calculated between all features and all objects
// linkageType to determine the type of linkage in hierarchical clustering
linkageType = 'single'
// dictionary to keep track of the nodes and clusters being formed
clusterKeeper = new dictionary
//to keep track of iterations
iterationCounter = 0
While Len(distanceMat) > 1:
    // find indices of the minimum distance in distanceMat
    ind1, ind2 = return minimum distance in distanceMat
    if linkageType == 'single':
        newRow = min(distanceMat[ind1, ind2])
    ElseIf linkageType == 'complete':
        newRow = max(distanceMat[ind1, ind2])
    ElseIf linkageType == 'average':
```

```

        newRow = mean(distanceMat[ind1, ind2])
//update distanceMat
// remove row and column with the minimum distance from distanceMat
distanceMat = distanceMat.drop([ind1, ind2])
// add the new clustered row in the distanceMat
distanceMat.append(newRow)

// update clusterKeeper dictionary
if ind1 or ind2 in clusterKeeper.keys(): //meaning that they are a pre-existing
clusters
    clusterKeeper[f"cluster {iterationCounter}"] =
{'node1':clusterKeeper[indPop1]['fullNodes'].copy() ,
"node2":clusterKeeper[indPop2]['fullNodes'].copy()}
    ElseIf ind1 in clusterKeeper.keys(): //ind1 is a previously known cluster
        clusterKeeper[f"cluster {iterationCounter}"] =
{'node1':clusterKeeper[indPop1]['fullNodes'].copy() , "node2":indPop2}
    ElseIf ind2 in clusterKeeper.keys(): //ind1 is a previously known cluster
        clusterKeeper[f"cluster {iterationCounter}"] =
{'node2':clusterKeeper[indPop2]['fullNodes'].copy() , "node1":indPop1}
        clusterKeeper["fullNodes"] = [clusterKeeper["node1"] , clusterKeeper["node2"]]
        iterationCounter++

```

2.3.3 DBSCAN

Input:

Dataset $D = [x_1, x_2, x_3, \dots, x_m]$;

Radius: the radius of search circle to determine a core point;

MinPts: the minimum number of points in the search circle to determine a core point.

Process:

Initialize the set of core points $\Omega = \Phi$

Generate kdtree with all points

For $j = 1, 2, \dots, m$ do

 If $|x_j$'s neighbors| \geq MinPts:

 Add x_j to core point set

 End if

Initialize the number of clusters $k = 0$

Initialize the set of unprocessed point $\Gamma = D$

While $\Omega \neq \Phi$ do:

 Initialize the set of one cluster $\gamma = \emptyset$

 Randomly select a o that $o \in \Omega$

 Initialize queue = { o }

$\Gamma = \Gamma \setminus \{o\}$

 While $Q \neq \Phi$ do:

 Pop out q from queue

 If $|q$'s neighbors| \geq MinPts:

 Let $\Delta = q$'s neighbors $\cap \Gamma$

 Add Δ to queue

$\Gamma = \Gamma \setminus \Delta$

 delete Δ from Ω if it's in Ω

 Add Δ to cluster γ

 End if

```

End while
k = k+1
Ck = γ
End while
Output:
Clusters C = [C1, C2, ..., Ck]

```

2.4 Validation

Two input vectors keep information about two different partitions (let say P and P') of the same dataset. We refer to a pair of points (xi, xj) (we assume that $i < j$) from the dataset using the following terms:

term	definition
SS	The number of pairs where both objects belong to the same cluster in both partitionings.
SD	The number of pairs where both objects belong to the same cluster in partitioning P but in P' do not.
DS	The number of pairs where in partitioning P both objects belong to different clusters, but in P' do not.
DD	The number of pairs where both objects belong to different clusters in both partitionings.

Table 2.3 : *Four possibilities of each pair*

The external indexes [2][3] are:

Index name	formula
Rand statistic	$R = (SS + DD)/(SS+SD+DS+DD)$
Jaccard coefficient	$J = SS/(SS + SD + DS)$
Folkes and Mallows index	$FM = \sqrt{SS/(SS + SD)} * \sqrt{SS/(SS + DS)}$

Table 2.4: *External indexes for validation*

3 Experiment

From external indexes, we can see that K-means have the best performance, while hierarchical algorithms and DBScan's performances are relatively poor.

	Jaccard Coefficient	FM Index	Rand Statistic
K-means	0.453	0.623	0.530
hierarchy	0.123	0.261	0.656

<i>DBScan</i>	<i>0.113</i>	<i>0.288</i>	<i>0.673</i>
---------------	--------------	--------------	--------------

Table 2.5: External validation indexes of 3 clustering methods

(1) K-means

From the experimentation, we see that the results obtained are the best for k-means clustering algorithm, despite the fact that trees and buildings are assigned to the same cluster.

(2) hierarchical clustering

In the case of hierarchical clustering, it is observed that since the clustering occurs on a binary basis where two objects or clusters are stacked together into a new cluster, sometimes it can happen that the clusters formed are single point clusters, thus having a single label associated with it. Moreover, the hierarchical clustering algorithm plots nearly all points to one cluster, and assigns them the same label.

(3) DBScan

DBScan's external indexes are similar to hierarchical methods, but we read each cluster manually, since an automated cluster reading approach would have costed us more time, it was more feasible to assign the automatically generated clusters to variables.

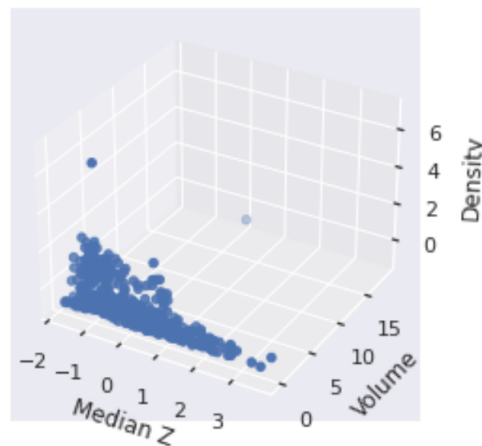


Figure 3.1: A plot of volume vs density vs median z values for given objects

4 Conclusion

Our findings:

1. The quality of the clusters can be very poor.
2. The quality of the clustering is highly affected by the quality of the dataset as well as the selected features, which affect the quality of the clustering significantly.
3. Logically simple clustering algorithms do not necessarily perform poorly, such as k-means.

To achieve a better result, we think the following could be done:

1. Extract more valuable features, like mean roughness. In this case, trees and buildings are hard to distinguish, but our classmate Max van Schendel's experiment shows that trees are very rough and buildings are very smooth.[4]

2. Add weight to key features, like height. Of course, the weight should be decided by experiment. In our experiment, we found that height can be one of the most distinguishing factors.
3. Integrate some Intelligent algorithms to determine parameters. For example, the ant colony algorithm.
4. Better feature selection would also lead to better results, additionally we can also look at running the classification algorithm on a binary basis to distinguish one object from the rest and not multiple objects at the same instant, this way a better ruleset for different objects could be found, however this would also make the algorithm further away from unsupervised clustering and closer to supervised clustering.

5 Work distribution

Name	Work in code	Work in report
Pratyush	Load dataset & make features Hierarchical clustering	Full cooperation
Cynthia	K-means clustering DBScan clustering Validation	

Table 5.1: *Work distribution in the assignment*

References

- [1] Zhihua Zhou. Machine Learning. Tsinghua University Press. 2016, 03(No.21):93-93.
- [2] Lukasz Nieweglowski. Standard External Measures: Rand index, Jaccard coefficient etc..
https://search.r-project.org/CRAN/refmans/clv/html/standard_external_measures.html
- [3] G. Saporta and G. Youness. *Comparing two partitions: Some Proposals and Experiments*.
<http://cedric.cnam.fr/PUBLIS/RC405.pdf>
- [4] Roughness document of CloudCompare.
<https://www.cloudcompare.org/doc/wiki/index.php?title=Roughness>