# Non-Supervised Clustering

Mariajose Franco Orozco
mfrancoo@eafit.edu.co
*EAFIT University*
*Mathematical Engineering*
Medellín, Colombia

## I. Introduction

Non-supervised clustering, also known as unsupervised clustering, is a machine-learning technique that consists of grouping data points into clusters based on their similarities. What makes it unsupervised is that there is no need for prior knowledge or labels as in supervised learning. This is because non-supervised clustering aims to identify patterns and structures within the dataset [1].

In the actual world, it is common to deal with huge volumes of data which include images, videos, text, and more. Organizing such data into rational groups is important in order to analyze these data. Clustering techniques are useful for completing this task of organizing data into different groups [2].

There exist many clustering algorithms, but in this paper, were performed 5 of them: Mountain Clustering, Subtractive Clustering, K-means Clustering, Fuzzy C-means, and DBSCAN Clustering. Clustering algorithms tend to use different similarity metrics, in this project 4 metrics were implemented: Euclidean, Mahalanobis, Manhattan, and Cosine distance. These metrics determine the similarity between the data points, and the algorithms determine the clusters according to these similarities [1].

In recent years, there has been a growing interest in non-supervised clustering. Because of this, it has a lot of applications fields, for example, data mining, pattern recognition, image analysis, bioinformatics, and more [3].

## II. Theoretical Framework

This section will explain the theory of the concepts used in this project.

### A. Metrics

4 metrics were considered: euclidean, mahalanobis, manhattan, and cosine distances. As it was mentioned above, these metrics are in charge of determining the similarity between data points.

*1) Euclidean distance:*

$$d(A,B) = \sqrt{\sum_{i=1}^{d}(A_i - B_i)^2} \tag{1}$$

where $d$ is the dimension of the space.

*2) Mahalanobis distance:*

$$d(A,B) = \sqrt{(A-B)C^{-1}(A-B)^T} \tag{2}$$

where $C$ refers to the covariance, and $C^{-1}$ is the inverse covariance.

*3) Manhattan distance:*

$$d(A,B) = \sum_{i=1}^{d} |A_i - B_i| \tag{3}$$

where $d$ is the dimension of the space.

*4) Cosine distance:*

$$d(A,B) = 1 - \frac{\sum_{i=1}^{d} A_i B_i}{\sqrt{\sum_{i=1}^{d} A_i^2 \sum_{i=1}^{d} B_i^2}} \tag{4}$$

where $d$ is the dimension of the space.

### B. Clustering Algorithms

Also, 5 clustering algorithms were implemented as well: mountain, subtractive, k-means, fuzzy c-means, and DBSCAN clustering algorithms.

*1)* ***Mountain Clustering****:* This algorithm consists of partitioning the space into a grid, where the intersections will be the candidates for the centroids. These candidates to be centroids will be denoted as $v$. Then, a mountain function is calculated for every candidate for centroid. This function represents a density measure for the data and is calculated as follows:

$$m(v) = \sum_{i=1}^{N} \exp\left(-\frac{||v - x_i||^2}{2\sigma^2}\right) \tag{5}$$

After calculating the mountain function for every grid intersection, the centroids will be selected. The first centroid will be the one with the highest density value of the mountain function.
For selecting the next centroids, the effect of the last centroid has to be eliminated, for this, a new equation will be calculated:

$$m_{new}(v) = m(v) - m(c_1)\exp\left(-\frac{||v - c_1||^2}{2\beta}\right) \tag{6}$$

where $\beta$ is a parameter.

The next centroid will be the one with higher density on this new mountain function. This process will continue until reaching the stop criteria, which is that, when the algorithm finds a centroid that is already in the list of centroids, it will stop and return the sufficient clusters found.
The dimension of the input data will increase the computation of this algorithm given that the mountain function has to be calculated for every grid point.

*2) **Subtractive Clustering**:* This algorithm is similar to the mountain algorithm, but the initial candidates for centroids will not be the grid intersections but the data points. This initialization decreases the computation needed in the mountain algorithm, which makes the subtractive algorithm more efficient.
The density function for every data point will be calculated as follows:

$$D(i) = \sum_{j=1}^{n} \exp\left(-\frac{||x_i - x_j||^2}{(r_a/2)^2}\right) \tag{7}$$

where $r_a$ is a parameter that represents a neighbor radius.

After calculating the density function for every data point, the centroids will be selected. The first centroid will be the one with the highest density value. A point will have a high-density value when it has more neighbor data points.

Then, the effect of the first centroid has to be eliminated and it is done with the following equation for the next density values:

$$D(i) = D(i) - D(c_1)\exp\left(-\frac{||x_i - x_{c1}||^2}{(r_b/2)^2}\right) \tag{8}$$

where $r_b$ is a parameter that represents a neighbor radius with density measure reductions, which means that the data points closer to the first centroid $c_1$, will have a higher reduction in their density measure. The next centroid will be the one with a higher density measure. This process will continue until reaching the stop criteria, which is that, when the algorithm finds a centroid that is already in the list of centroids, it will stop and return the sufficient clusters found.

*3) **K-Means Clustering**:* The algorithm of K-Means is useful for finding clusters data in which the cost function will be minimized. This algorithm receives the number of clusters that has to be found in the dataset. The process starts by initializing the centroids as random points. Then, the membership matrix will be calculated with the following equation:

$$u_{ij} = \begin{cases} 1 & ||x_j - c_i||^2 \leq ||x_j - c_k||^2 \forall k \neq i \\ 0 & otherwise \end{cases} \tag{9}$$

In this membership matrix, $u_{ij} = 1$ when $x_j$ belong to the $i^{th}$ group.

Then, the cost function has to be calculated which is:

$$J = \sum_{i=1}^{c} J(i) = \sum_{i=1}^{c}\left(\sum_{k} ||x_k - c_i||^2\right) \tag{10}$$

The algorithm will break if there is no improvement in the solution. Then, the centroids have to be actualized with the function

$$c_i = \frac{\sum_k x_k}{\sum_{j=1}^{n} u_{ij}} \tag{11}$$

This process will continue until reaching the stop criteria.

*4) **Fuzzy C-Means Clustering**:* In the K-Means algorithm, a data point belongs to a group or not, but in Fuzzy C-Means, the data point can belong to different groups or clusters with a certain membership grade between 0 and 1. This algorithm also tries to find the clusters for a dataset minimizing a cost function.

The first step is to initialize the membership matrix with random values between 0 and 1 guaranteeing that

$$\sum_{i=1}^{c} u_{ij} = 1, \forall j = 1, 2, ..., n \tag{12}$$

Then, the fuzzy cluster centroids will be calculated using the following function:

$$c_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m} \tag{13}$$

After this, the following cost function will be calculated

$$J = \sum_{i=1}^{c} J(i) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^m d_{ij}^2 \tag{14}$$

where $c_i$ is the fuzzy cluster centroid of the $i^{th}$ group, $d_{ij} = ||c_i - x_j||$ and $m \in [0, \infty)$ .

The algorithm will break if there is no improvement in the solution. If it does not break, an update to the membership matrix has to be made using the following equation

$$u_{ij} = \frac{1}{\sum_{k=1}^{c}\left(\frac{d_{ij}}{d_{kj}}\right)^{2/(m-1)}} \tag{15}$$

This process will continue until reaching the stop criteria.

*5) **DBSCAN CLustering**:* For the DBSCAN algorithm, the first step is to initialize a vector of zeros that will represent the data points already visited. It initializes in zeros because no data point has been visited at the beginning.

The algorithm receives 2 parameters: epsilon and minPts. The epsilon is the radius of every neighborhood, and minPts is the minimum number of neighbors accepted for the neighborhoods. The next step is to find the neighbors for each data point and marked them as visited. These are the data points that are within the epsilon radius. The clusters will be created considering the neighbors of every data point.

## III. Methodology

First, a toy dataset was used for analyzing in 2D and 3D how the algorithms worked. The toy dataset considered was the Iris dataset, and a validation of the clusters was performed using intra and extra cluster validation. The score performed for the intra cluster validation was the silhouette score, and for the extra cluster validation was the accuracy.

Then, the real dataset used in this implementation was the *Rice Dataset Cammeo and Osmancik* which can be downloaded from **here**. It consists of a study made in 2019 by [4] in which 2 types of rice: Cammeo and Osmancik, were analyzed and some characteristics were extracted from each type of them resulting in 3810 data and 7 attributes for each one of them [5]. The attributes are:

- **Area:** Returns the number of pixels within the boundaries of the rice grain.
- **Perimeter:** Calculates the circumference by calculating the distance between pixels around the boundaries of the rice grain.
- **Major Axis Length:** The longest line that can be drawn on the rice grain, i.e. the main axis distance, gives.
- **Minor Axis Length:** The shortest line that can be drawn on the rice grain, i.e. the small axis distance, gives.
- **Eccentricity:** It measures how round the ellipse, which has the same moments as the rice grain, is.
- **Convex Area:** Returns the pixel count of the smallest convex shell of the region formed by the rice grain.
- **Extent:** Returns the ratio of the region formed by the rice grain to the bounding box pixels.

This dataset is very large and because of the high computation of the algorithms implemented, I considered only 4 attributes: area, perimeter, major axis length and minor axis length, and 300 datapoints.

This way, the resultant dataset consists of 4 inputs of 300 data, all of them numerical variables, ready to be normalized.

As we are working with unsupervised algorithms, the outputs are not going to be considered for this project.

5 clustering algorithms were performed and some experimentation was made in order to see the effect of the hyperparameters on the output.

Then, the idea was to extend and reduce the dimension of the problem and apply the clustering algorithms to these new problems. For extending the dimension of the dataset, an autoencoder was used and the high-dimension dataset resultant is 5 dimensional. For the reduction of the space, the UMAP algorithm was implemented and the low-dimension dataset resultant is 2 dimensional. Finally, the 5 clustering algorithms were also applied to these new datasets.

## IV. Experimentation

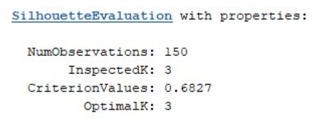
### A. *Toy Dataset*

The first step performed was using the intra and extra cluster validation with the toy dataset in order to verify the performance of the algorithms implemented.

For the intra cluster evaluation, the Silhouette criterion was performed. This function receives the clusters obtained with every algorithm and the output is an index between -1 and 1, which determines how well is the data classified. If the index is near 1, it means that the clustering is well done.

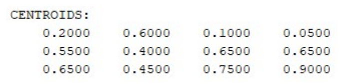
For the extra cluster validation, the accuracy of the output obtained with every algorithm was performed. The results obtained for every algorithm are presented in the following sections.

*1) **Mountain Clustering**:* For this experiment, the parameters used were sigma = 0.1, beta = 0.4 and gr = 20. The Silhouette index obtained was **0.6827**.



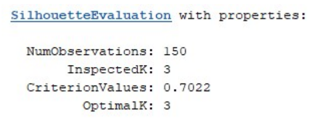
```
SilhouetteEvaluation with properties:

    NumObservations: 150
         InspectedK: 3
    CriterionValues: 0.6827
           OptimalK: 3
```

The accuracy obtained was **86.6667%**, with 3 clusters found and the centroids were



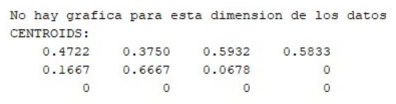
```
CENTROIDS:
    0.2000    0.6000    0.1000    0.0500
    0.5500    0.4000    0.6500    0.6500
    0.6500    0.4500    0.7500    0.9000
```

*2) **Subtractive Clustering**:* For this experiment, the parameters used were ra = 0.8 and rb = 2*ra. The Silhouette index obtained was **0.7022**.



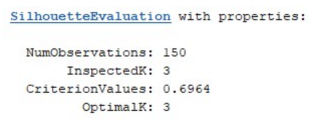
```
SilhouetteEvaluation with properties:

    NumObservations: 150
         InspectedK: 3
    CriterionValues: 0.7022
           OptimalK: 3
```

The accuracy obtained was **66%**, with 3 clusters found and the centroids were



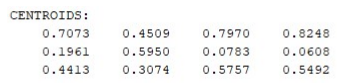
```
No hay grafica para esta dimension de los datos
CENTROIDS:
    0.4722    0.3750    0.5932    0.5833
    0.1667    0.6667    0.0678         0
         0         0         0         0
```

*3) **K-Means Clustering**:* For this experiment, the parameters used were the number of clusters, nc = 3 and delta = 0.00001. The Silhouette index obtained was **0.6964**.



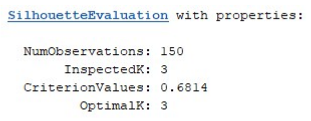
```
SilhouetteEvaluation with properties:

    NumObservations: 150
         InspectedK: 3
    CriterionValues: 0.6964
           OptimalK: 3
```

The accuracy obtained was **88.6667%** and the centroids were



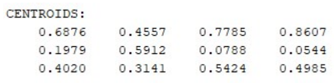
```
CENTROIDS:
    0.7073    0.4509    0.7970    0.8248
    0.1961    0.5950    0.0783    0.0608
    0.4413    0.3074    0.5757    0.5492
```

*4) **Fuzzy C-Means Clustering**:* For this experiment, the parameters used were the number of clusters, nc = 3, m = 7 and delta = 0.00001. The Silhouette index obtained was **0.6814**.



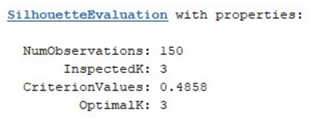
```
SilhouetteEvaluation with properties:

  NumObservations: 150
       InspectedK: 3
  CriterionValues: 0.6814
         OptimalK: 3
```

The accuracy obtained was **90%** and the centroids were



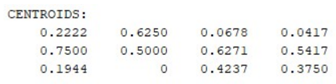
```
CENTROIDS:
    0.6876    0.4557    0.7785    0.8607
    0.1979    0.5912    0.0788    0.0544
    0.4020    0.3141    0.5424    0.4985
```

*5) **DBSCAN Clustering**:* For this experiment, the parameters used were eps = 0.75 and minPts = 10. The Silhouette index obtained was **0.4858**.



```
SilhouetteEvaluation with properties:

  NumObservations: 150
       InspectedK: 3
  CriterionValues: 0.4858
         OptimalK: 3
```

The accuracy obtained was **55.3333%** and the centroids were



```
CENTROIDS:
    0.2222    0.6250    0.0678    0.0417
    0.7500    0.5000    0.6271    0.5417
    0.1944         0    0.4237    0.3750
```

Finally, with the intra and extra cluster evaluation scores, can be concluded that the algorithms presents good results for the toy dataset. It can be seen that the centroids obtained with K-Means and Fuzzy C-Means are very similar and also, were the algorithms with better accuracy. The worst performance is the obtained with the DBSCAN algorithm, in which it was obtained the lower accuracy and the lower Silhouette index.

## B. *Real Dataset*

For the experimentation, a change in the hyperparameters was performed in order to see how they affect the results obtained and which ones are the better ones for the project.

*1) **Mountain Clustering**:* For the mountain algorithm, a variation in sigma and beta was performed. Sigma was considered as $[0.1, 0.3, 0.5]$ and beta as $[0.2, 0.4, 0.6]$, and all the possible combinations were considered and are presented in the following table.

| Mountain Clustering Original Dataset | | | |
|---|---|---|---|
| | Hyperparameters combination sigma & beta | | |
| | **0.1 & 0.2** | **0.1 & 0.4** | **0.1 & 0.6** |
| **Silhouette score** | 0.4088 | 0.5861 | 0.7462 |
| **Optimal K** | 10 | 3 | 2 |
| | **0.3 & 0.2** | **0.3 & 0.4** | **0.3 & 0.6** |
| **Silhouette score** | 0.1110 | 0.4523 | 0.6956 |
| **Optimal K** | 16 | 4 | 2 |
| | **0.5 & 0.2** | **0.5 & 0.4** | **0.5 & 0.6** |
| **Silhouette score** | 0.0821 | 0.3163 | 0.3158 |
| **Optimal K** | 16 | 6 | 4 |

It can be seen that differents numbers of clusters were obtained, but the models with the best score are the obtained with 2 clusters, with a silhouette score of 0.7462 and 0.6959. The model that will be considered for obatining the results with the dataset in high and low dimension is sigma = 0.1 and beta = 0.6, which was the model with higher silhouette score.

*2) **Subtractive Clustering**:* For the subtractive algorithm, a variation in ra and rb was performed. ra was considered as $[0.6, 0.8, 1]$ and beta as $[1 * ra, 1.5 * ra, 2 * ra]$, and all the possible combinations were considered and are presented in the following table.

| Subtractive Clustering Original Dataset | | | |
|---|---|---|---|
| | Hyperparameters combination ra & rb | | |
| | **0.6 & 1*ra** | **0.6 & 1.5*ra** | **0.6 & 2*ra** |
| **Silhouette score** | 0.4629 | 0.5475 | 0.7226 |
| **Optimal K** | 6 | 3 | 2 |
| | **0.8 & 1*ra** | **0.8 & 1.5*ra** | **0.8 & 2*ra** |
| **Silhouette score** | 0.5662 | 0.5581 | 0.6952 |
| **Optimal K** | 3 | 3 | 2 |
| | **1 & 1*ra** | **1 & 1.5*ra** | **1 & 2*ra** |
| **Silhouette score** | 0.5262 | 0.6952 | 0.6568 |
| **Optimal K** | 3 | 2 | 2 |

The most recurrent number of clusters obtained were 2 and 3 clusters, but it is evident that every model with 2 clusters have a better silhouette score than the models with 3 clusters. The model that will be considered for obatining the results with the dataset in high and low dimension is ra = 0.6 and rb = 2*ra, which was the model with higher silhouette score.

*3) **K-Means Clustering**:* For the k-means algorithm, a variation in the number of clusters was performed. An experimentation with $[1, 2, 3, 4, 5, 6, 7, 8, 9]$ clusters were made in order to see which number of clusters were the better fit considering the silhouette score.

| K-Means Clustering Original Dataset | | | |
|---|---|---|---|
| | Hyperparameters combination # clusters | | |
| | **1** | **2** | **3** |
| **Silhouette score** | NAN | 0.7588 | 0.6048 |
| | **4** | **5** | **6** |
| **Silhouette score** | 0.5799 | 0.5141 | 0.5392 |
| | **7** | **8** | **9** |
| **Silhouette score** | 0.4989 | 0.4679 | 0.4375 |

Is possible to evidence in the table that the silhouette score increments when the number of clusters is lower and decreases when considering more number of clusters. The best score obtained was with 2 clusters with a silhouette score of 0.7588, and the worst score was obtained with 9 clusters. As it can be seen in the results obtained with all the algorithms above, the best score is always with 2 clusters, which allows us to determine that our data is distributed that way.

*4) **Fuzzy C-Means Clustering**:* For the fuzzy c-means algorithm, a variation in the number of clusters and the m parameter was performed. The number of clusters considered for the experimentation were $[1, 2, 3]$ and the variation in the m exponent parameter were $[3, 5, 7]$. All the combinations were considered and are presented in the following table.

| Fuzzy C-Means Clustering Original Dataset | | | |
|---|---|---|---|
| | Hyperparameters combination # clusters & m | | |
| | **1 & 3** | **1 & 5** | **1 & 7** |
| **Silhouette score** | NAN | NAN | NAN |
| | **2 & 3** | **2 & 5** | **2 & 7** |
| **Silhouette score** | 0.7084 | 0.7073 | 0.7073 |
| | **3 & 3** | **3 & 5** | **3 & 7** |
| **Silhouette score** | 0.5947 | 0.5958 | 0.5960 |

When considering only 1 cluster, there is no silhouette score score obtained because if it exists only one cluster in the data, there is no way to determine if the data is not well classified. The higher silhouette score obtained was also with 2 clusters and an m exponent of 3. This is the model that will be considered for the high and low dimension datasets.

*5) **DBSCAN Clustering**:* For the DBSCAN algorithm, a variation in the radius parameter (eps) and the minimum of points per cluster parameter (minPts) was performed. The variation in the eps considered for the experimentation was $[0.25, 0.5, 0.75]$ and the variation in the minPts parameter was $[10, 20, 30]$. All the combinations were considered and are presented in the following table.

| DBSCAN Clustering Original Dataset | | | |
|---|---|---|---|
| | Hyperparameters combination eps & minPts | | |
| | **0.25 & 10** | **0.25 & 20** | **0.25 & 30** |
| **Silhouette score** | 0.2130 | 0.3445 | 0.3498 |
| **Optimal K** | 14 | 11 | 8 |
| | **0.5 & 10** | **0.5 & 20** | **0.5 & 30** |
| **Silhouette score** | 0.5675 | 0.5470 | 0.5432 |
| **Optimal K** | 3 | 3 | 3 |
| | **0.75 & 10** | **0.75 & 20** | **0.75 & 30** |
| **Silhouette score** | 0.6780 | 0.7021 | 0.6920 |
| **Optimal K** | 2 | 2 | 2 |

It can be seen that differents numbers of clusters were obtained, and the most recurrent number of clusters obtained were 2 and 3 clusters as in the experimenation with the subtractive algorithm. The models with the best score are the obtained with 2 clusters as in all the algorithms implemented. The model that will be considered for obtaining the results with the dataset in high and low dimension is eps = 0.75 and minPts = 20, which was the model with higher silhouette score, 0.7021.

## V. Results

After defining which hyperparameters will be used for every algorithm, the algorithms were performed with the original dataset, the high dimension dataset and the low dimension dataset. The high dimension dataset considered is 5 dimensional and was obtained with an autoencoder. The low dimension dataset is 2 dimensional and was obtained with the Umap algorithm. The results obtained for every algorithm are presented in the following tables.

| Mountain Clustering | | | | |
|---|---|---|---|---|
| | **Dimension** | **Silhouette** | **Optimal K** | **Centroids** |
| **Original** | 4 | 0.7010 | 2 | [0.4000 0.4000 0.3000 0.5500]<br>[0.7500 0.8000 0.7500 0.7000] |
| **High Dim** | 5 | 0.5263 | 3 | [0.7500 0.2000 0.8500 0.1500 0.8000]<br>[0.2500 0.5000 0.2000 0.8000 0.2000]<br>[0.7500 0.6000 0.3500 0.5500 0.3000] |
| **Low Dim** | 2 | 0.5094 | 2 | [0.7500 0.1500]<br>[0.2000 0.5000] |

| Subtractive Clustering | | | | |
|---|---|---|---|---|
| | **Dimension** | **Silhouette** | **Optimal K** | **Centroids** |
| **Original** | 4 | 0.6847 | 2 | [0.4355 0.3896 0.3409 0.5614]<br>[0.7799 0.8223 0.7740 0.7185] |
| **High Dim** | 5 | 0.4593 | 3 | [0.6337 0.5035 0.4451 0.4961 0.4243]<br>[0.6771 0.2380 0.8060 0.1861 0.7705]<br>[0.2235 0.5067 0.1870 0.8054 0.1925] |
| **Low Dim** | 2 | 0.5068 | 2 | [0.6490 0.3910]<br>[0.1798 0.4711] |

| K-Means Clustering | | | | |
|---|---|---|---|---|
| | **Dimension** | **Silhouette** | **Optimal K** | **Centroids** |
| **Original** | 4 | 0.7085 | 2 | [0.7058 0.7389 0.6872 0.6843]<br>[0.3715 0.3745 0.2907 0.5132] |
| **High Dim** | 5 | 0.5331 | 3 | [0.5250 0.2464 0.7284 0.2800 0.7047]<br>[0.7060 0.5553 0.4214 0.5066 0.3916]<br>[0.3192 0.4878 0.2647 0.7232 0.2590] |
| **Low Dim** | 2 | 0.5263 | 2 | [0.3013 0.5207]<br>[0.7604 0.3845] |

| Fuzzy C-Means Clustering | | | | |
|---|---|---|---|---|
| | **Dimension** | **Silhouette** | **Optimal K** | **Centroids** |
| **Original** | 4 | 0.7085 | 2 | [0.7284 0.7613 0.7106 0.6986]<br>[0.3578 0.3584 0.2732 0.5067] |
| **High Dim** | 5 | 0.5156 | 3 | [0.5831 0.5653 0.3030 0.6378 0.2767]<br>[0.2613 0.2528 0.6027 0.4314 0.6045]<br>[0.6993 0.2655 0.7818 0.2052 0.7451] |
| **Low Dim** | 2 | 0.5327 | 2 | [0.2625 0.4774]<br>[0.7907 0.4092] |

| DBSCAN Clustering | | | | |
|---|---|---|---|---|
| | **Dimension** | **Silhouette** | **Optimal K** | **Centroids** |
| **Original** | 4 | 0.6192 | 2 | [0.3347 0.2785 0.1930 0.5451]<br>[0.8866 0.8933 0.8253 0.8376] |
| **High Dim** | 5 | 0.1399 | 3 | [0.4278 0.5693 0.4414 0.5109 0.4654]<br>[0.7557 0.1768 0.8462 0.1485 0.7993]<br>[0.1206 0.7361 0 1.0000 0] |
| **Low Dim** | 2 | 0.3243 | 2 | [0.3195 0.1296]<br>[0.6586 0.9587] |

It can be seen in the tables presented above that when increasing the dimension of the space to 5 dimensions, the algorithms find one more cluster that the determined in the experimentations above, and when decreasing the dimension of the space to 2 dimensions, it finds 2 clusters as in the original dimension considered.

For the low dimension dataset, it was possible to obtain graphics that shows the distribution of the data and the clusters obtained with every model.

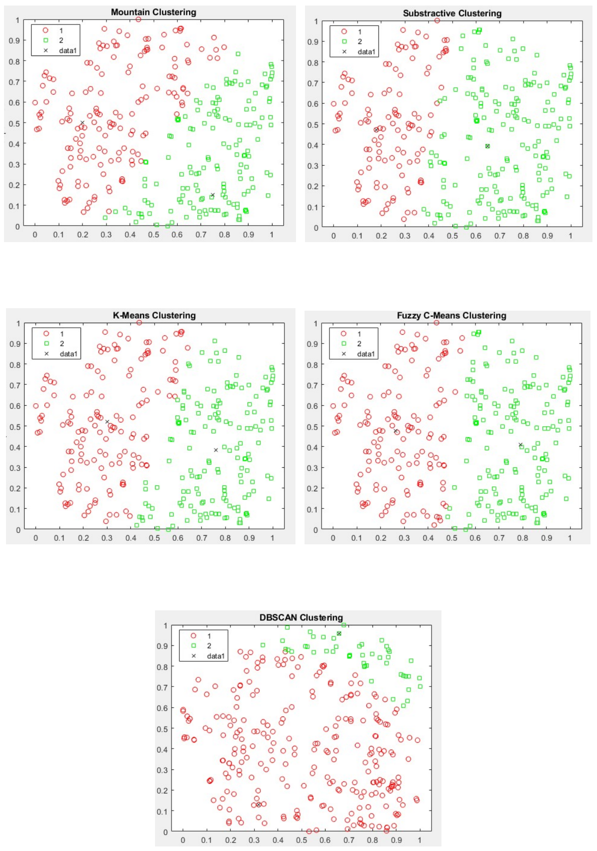


Fig. 1. Clusters obtained with the differents algorithms vs the real data clusters in low dimension

Is possible to evidence that the clusters obtained with the mountain, subtractive, k-means and fuzzy c-means algorithms are similar to each other and different from the results obtained with the DBSCAN algorithm, but also, the silhouette score obtained for this last algorithm is the worst of the scores compared to the other algorithms.

## VI. Conclusions

In sections IV and V we could see that the optimal number of clusters for the dataset used is 2 clusters. This results were consistent for every model and algorithm implemented. Also, we could see that when the algorithm suggested other number of clusters as the optimal, the silhouette score was lower, which means that the results obtained with the number of clusters suggested were not well classified.

For this project, the algorithms were implemented in matlab, but I had some problems when executing the mountain algorithm because of the high computation of the algorithm given that it has to calculate the mountain function in every intersection point of the grid. Because of this, the dataset needed to be less than 5 attributes. The dataset used was 4 dimensional, the high dimension was 5 dimensional, and the low dimension was 2 dimensional. For a future work, I would like to repeat this project with a dataset with higher dimensions in order to see the performance of the algorithms.

Finally, it can be concluded that the algorithms with the best performance were the k-means and the fuzzy c-means, but this algorithms receives the number of clusters to search for. When executing the algorithms that does not receive the number of clusters, the subtractive was the algorithm with the best performance and does not need high computation as mountain algorithm.

## References

[1] M. G. Omran, A. P. Engelbrecht, and A. Salman, “An overview of clustering methods,” *Intelligent Data Analysis*, vol. 11, no. 6, pp. 583–605, 2007.

[2] A. Ghosal, A. Nandy, A. K. Das, S. Goswami, and M. Panday, “A Short Review on Different Clustering Techniques and Their Applications,” in *Advances in Intelligent Systems and Computing*, 2020.

[3] L. V. Bijuraj, “Clustering and its Applications,” *Proceedings of National Conference on New Horizons in IT - NCNHIT 2013*, pp. 169–172, 2013.

[4] M. Koklu and I. Cinar, “Classification of Rice Varieties Using Artificial Intelligence Methods,” *International Journal of Intelligent Systems and Applications in Engineering*, 2019.

[5] U. M. L. Repository, “Rice (Cammeo and Osmancik) Data Set,” 2019. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Rice+{%}28Cammeo+and+Osmancik{%}29{#}