Clustering - Project

MSc Data Science and Information Technologies

Andrinopoulou Christina (ds2200013)

Contents

1 Data set

The data set is a 3-dimensional array. In particular, the size of the matrix is $150 \times 150 \times 204$ and corresponds to a picture with 150×150 pixels and each pixel is characterized by a vector with 204 elements. These elements are the spectral bands for a specific pixel. The picture depicts an area of the Salinas valley in California. As we can see in figure 1 there are different areas that correspond to different kinds of crops.

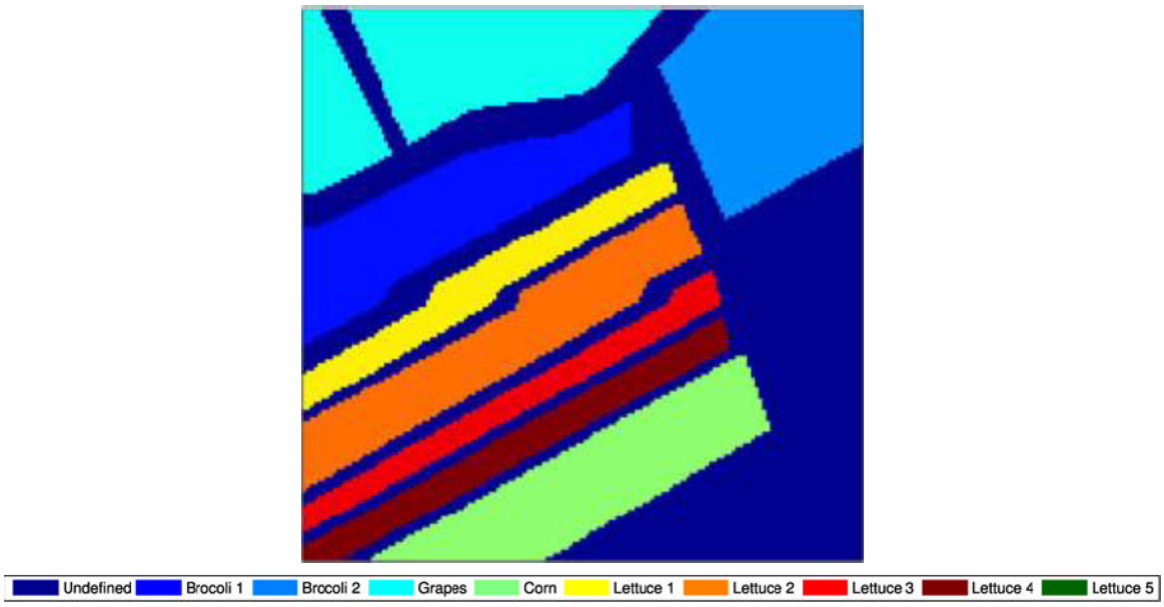

Figure 1: Salinas valley in California

Each pixel of this picture is a vector that contains the corresponding spectral bands. Therefore, the data set includes these pixels and the goal is to cluster the pixels based on the kind of crop. We should mention that we take into consideration only the pixels that belong to a class, which means that are part of a specific crop-area in the picture. So, we omit the pixels that belong to the dark blue area.

2 Pre-processing

Before executing different clustering algorithms, we should examine the data set and try to transform it, so as to show the physical clusters off.

As we have already discussed, each pixel is characterized by a vector, which contains 204 elements. So, the patterns of the data set have 204 features and as a result, they "live" in a 204-dimensional space. The dimension of the data set is big and this leads to a well-known problem, which is called the *curse of dimensionality*.

Generally, if the patterns of a data set have many features, they are scattered in a high-dimensional space. Let assume that the total number of the patterns in the data set is n. As the number of features increases, the same patterns are scattered to a higher and higher dimensional space and as a result, they do not represent the "rule", but the "exception". What we are trying to say is that the term "distance" between two patterns is degenerate. Therefore, if the distance between two patterns is d, this is not a clue about the similarity/dissimilarity between these patterns. However, the clustering is based on the distance measures and as a result, the curse of dimensionality has an extremely bad effect on the clustering.

We try to cope with this difficulty, using the *Principal Component Analysis* (PCA). PCA reduces the dimensionality of the data set and at the same time preserves as much information as possible. For the preserving of the information of the data set, PCA "compresses" the dimensions of the data set and creates new features that are a linear combination of the initial ones.

However, PCA creates a representative data set if the range of the features is approximately the same otherwise, it tends to discriminate in favor of the features that have larger range of values compared to others. Therefore, it is crucial the data set be normalized, before applying the PCA algorithm.

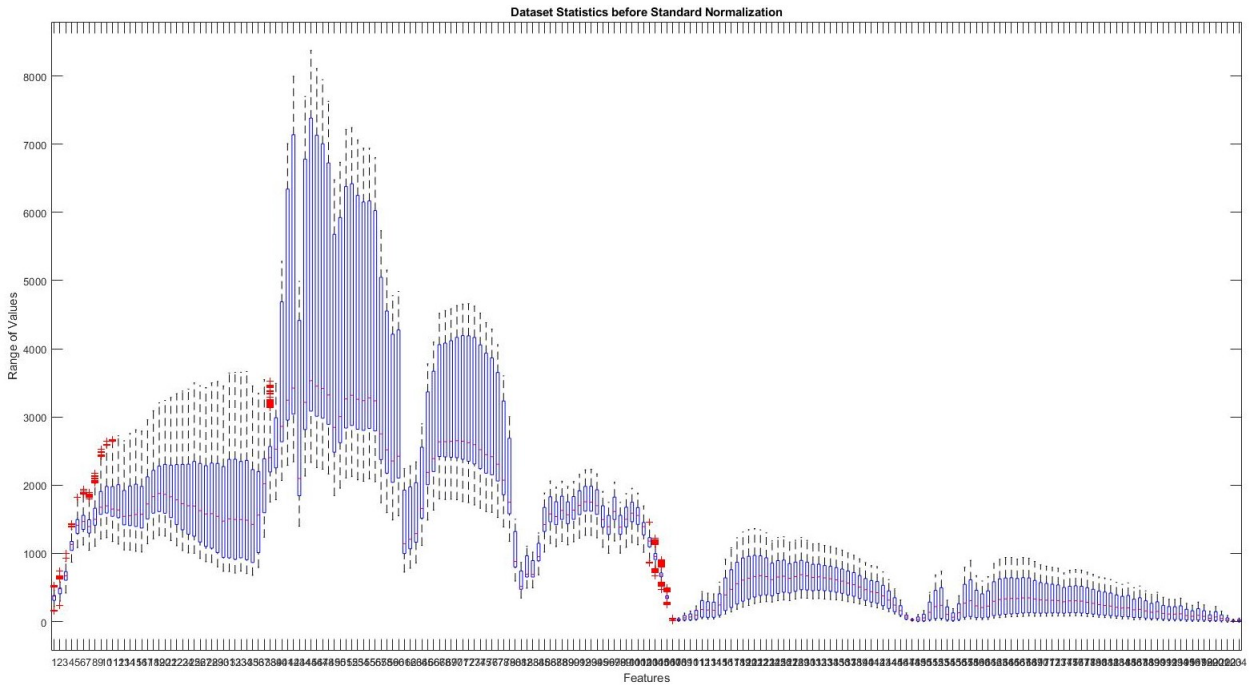

Figure 2: The boxplot per feature of the data set before applying the standard normalization algorithm

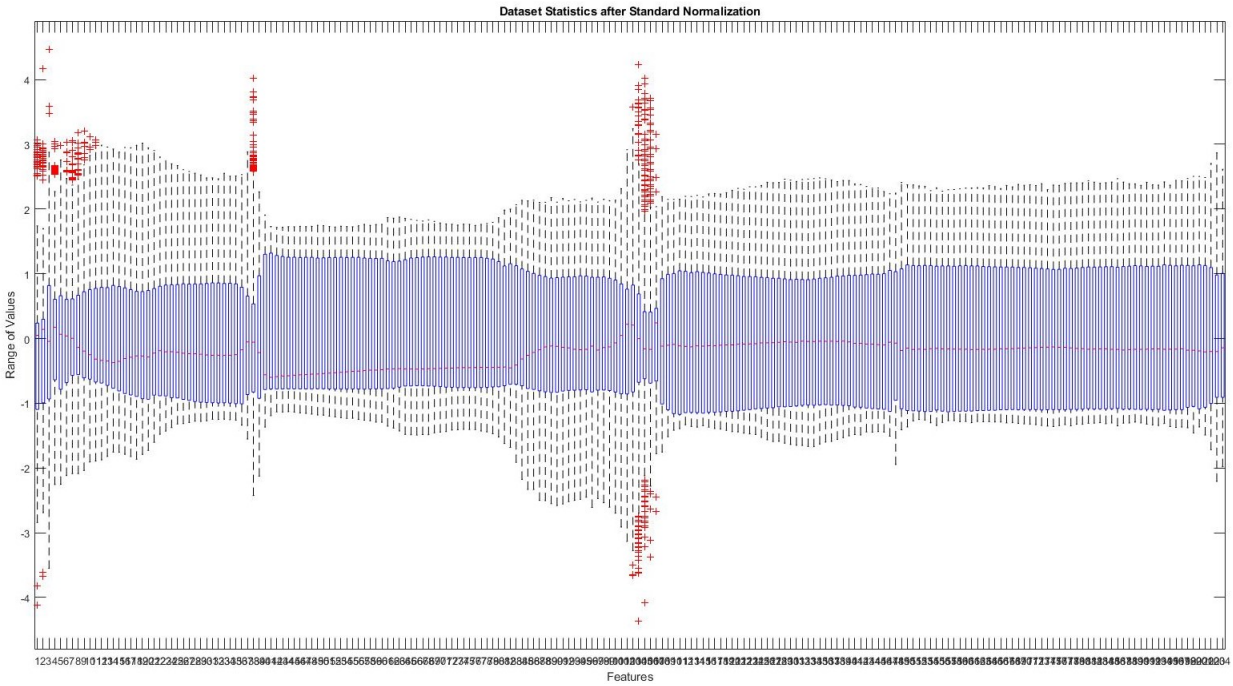

Figure 3: The boxplot per feature of the data set after applying the standard normalization algorithm

We select as normalization algorithm the *standard normalization*. This kind of normalization uses the mean and standard deviation of each feature. As we can see in figure 3, the algorithm collects the majority of the patterns (per feature) around the zero value. If a data point is equal to zero after the normalization, this point was equal to the mean value of the corresponding feature before the normalization. The range of the values of the features is approximately the same, so the data set, after the standard normalization, is suitable for PCA.

We apply the PCA algorithm to the data set and we create a new data one that is based on the PCA results, which contains only 3 features. As you can see in figure 6 the first two principal components summarize the majority of the information of the data set and this is the reason why we reduced the dimensionality of the data set into 3.

The data set after applying the PCA algorithm is shown in figures 4, 5.

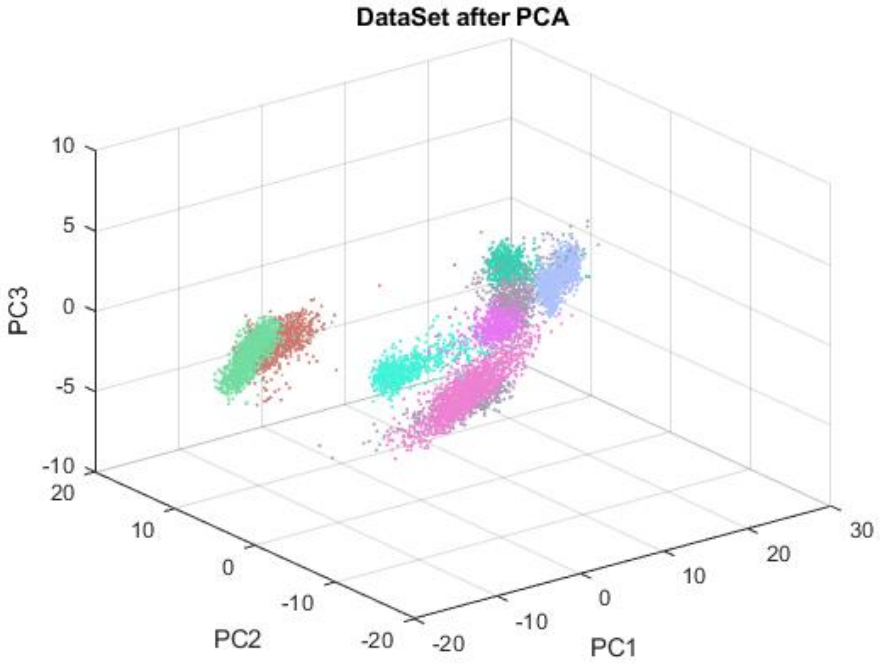

Figure 4: Data set after PCA. Each point is a pixel and each color refers to a different class. (3-dimensional case)

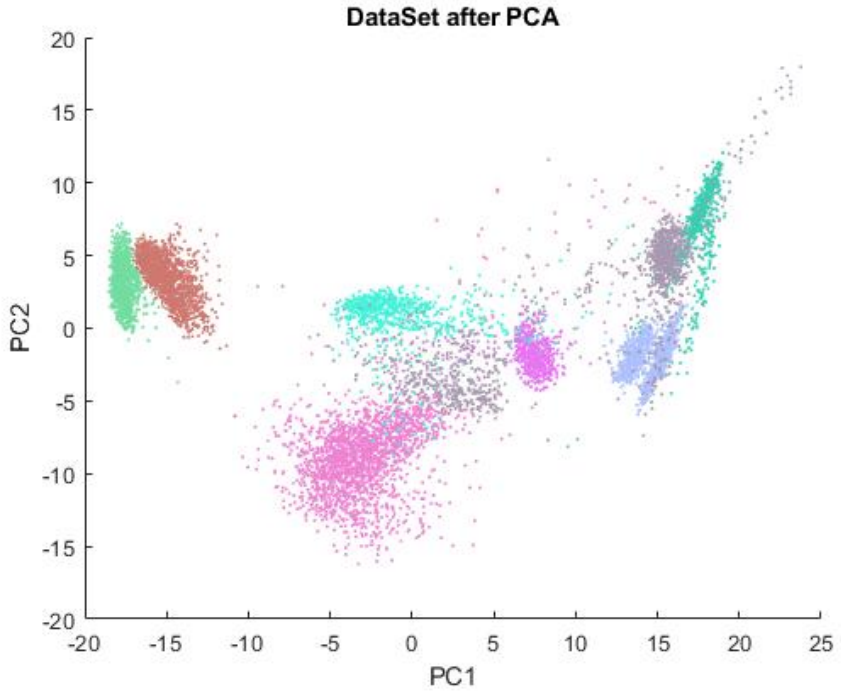

Figure 5: Data set after PCA. Each point is a pixel and each color refers to a different class. (2-dimensional case)

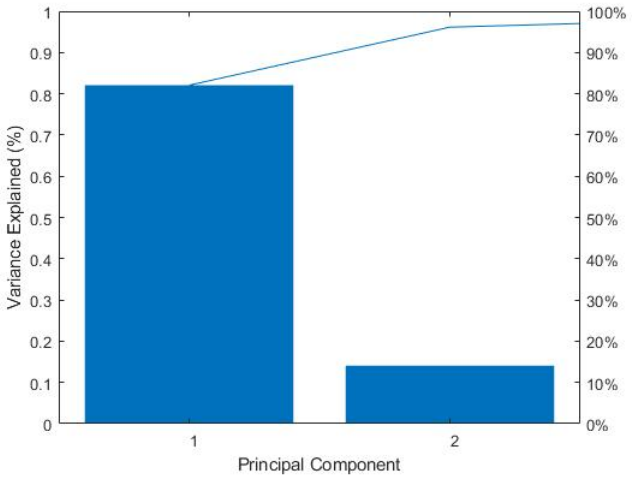

Figure 6: The initial information of the data set per principal component

3 Evaluation of clusterings

We are going to use many different techniques in order to evaluate the results of each clustering algorithm. First of all, we should mention that in this case we know the real labels of the data points, so we can compare the clusters that are produced by an algorithm with the real clusters. We can use the "valley image", so as to extract conclusions about the results. Also, we can use the *confusion matrix* for the labels. In a confusion matrix, each row corresponds to the instances that belong to the real class and each column corresponds to the instances that belong to the predicted class, or vice versa. Also, based on the confusion matrix, we can calculate the *accuracy* of the model, using the formula below:

$$accuracy = \frac{1}{N}\sum_{i=1}^{N} C(i,i)$$

if C is the confusion matrix and N is the total number of patterns. We should mention that we are not going to use the confusion matrix for more or less predicted labels than the real ones because this will not help us to make safe conclusions about the clustering results.

Furthermore, we are going to evaluate the clusters with techniques that do not require the knowledge of the real labels. We are going to use the *silhouette score*. This score estimates the similarity between a data point and the cluster that belongs to, in comparison to the other clusters. The range of the silhouette values is between -1 and 1. If a point has a silhouette score equal to one, it is assigned to the appropriate cluster and if a data point has a silhouette score equal to -1, it means that is assigned to the wrong cluster.

4 Cost Function Optimization Algorithms

A well-known family of clustering algorithms is the *cost function optimization algorithms*. The goal of these algorithms is to optimize a function J. This function expresses the cost of the clustering and is a function of the data points. The parameter of the function is a vector $\vec{\theta}$. This parameter may refer to the *representatives* of each cluster. A representative may be a point or a line and represents a cluster. We want to find the optimal set of the parameters. So, we want to find a vector $\vec{\theta}$ that minimizes the cost function J and that's why these algorithms are called cost function optimization algorithms.

There are four different kinds of algorithmic schemes in this concept and we are going to explain all of them and use them in order to cluster the data points. In all cases, we can use a different kind of representatives, but we prefer using point representatives because figure 4 is a strong indication that the clusters are compact and for compact clusters, the best choice for representatives is the point representatives.

4.1 Hard Clustering

The first approach that we are going to use in order to create clusters of the patterns is *hard clustering*. In this case, each pattern belongs to only one cluster, after the clustering procedure.

We are going to use point representatives for the clusters, as we have already discussed. The combination of the hard scheme with point representatives creates a very popular clustering algorithm, which is called *k-means*.

The goal of k-means is to move the representatives to the regions that correspond to clusters. First of all, the algorithm assigns each pattern to a cluster representative and after that, it updates the representatives. The recalculation of the cluster representatives is based on the mean value of the data points that belong to the corresponding cluster. Therefore, each time that the composition of the cluster changes, the representatives should be calculated again. These two steps are the core of the k-means and are repeated until the representatives do not change at all.

The major drawback of the algorithm is that it needs the number (k) of the clusters and some initial representatives in order to cluster the patterns. We utilize two different approaches in order to find the appropriate initial representatives for the algorithm.

4.1.1 Determination of number of clusters based on the Modified Basic Sequential Algorithmic Scheme

The first approach is based on another kind of clustering algorithm that belongs to the sequential clustering algorithms.

In the *base sequential algorithmic scheme* (BSAS) each data point is assigned to the closest cluster that has already been defined or if the distance between the data point and the closest cluster is greater than a threshold and the number of clusters is not bigger than a predefined number, the algorithm creates a new cluster.

However, we used a modification of this scheme, in order to find the appropriate initial representatives. In the basic algorithmic scheme, a data point is assigned to a cluster, before creating all the clusters. On the other hand, in the modified scheme, the first step is to define all the clusters of the data set and after that to assign the data points to them.

The modified scheme is executed many times for a range of distance thresholds. The number of clusters for each distance threshold is considered to be the most frequent number. The relationship between the distance threshold and the number of clusters is plotted and the most suitable number of clusters corresponds to a plateau in the graph.

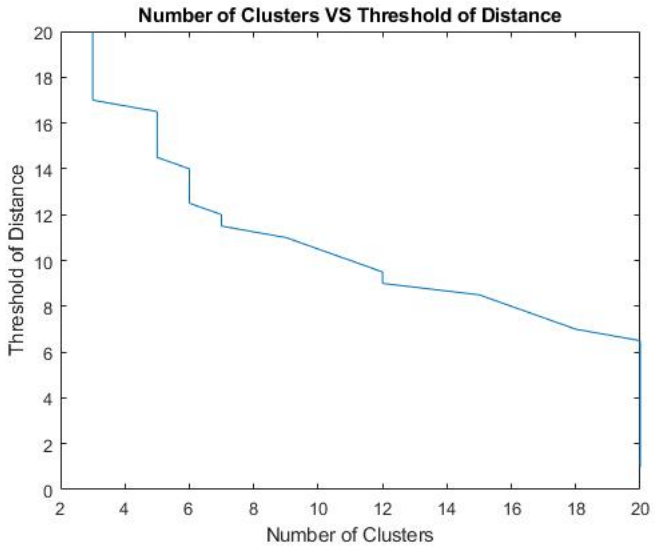

Figure 7: Relationship between the number of clusters and the distance threshold

Based on the above figure, we can conclude that the data set may contain 6 or 7 physical clusters. However, we are going to use another technique for the discovery of the appropriate number of clusters for the specific data set.

4.1.2 Determination of number of clusters based on the cost function of k-means

The second approach that was used in order to find the appropriate number of clusters for the k-means algorithm is based on a well-known technique of *cluster validity*. Cluster validity refers to the quantification of the evaluation of the clustering results. The cluster validity contains three categories of criteria: the *external criteria*, the *internal criteria* and the *relative criteria*. In this case, we are going to use only the last kind of criteria.

The goal is to find the optimal clustering from a range of different clusterings of the data set, based on a criterion. In this case, we check the different clusterings that come from the k-means, using a different number of clusters and different initial representatives and we evaluate each clustering based on the corresponding cost function.

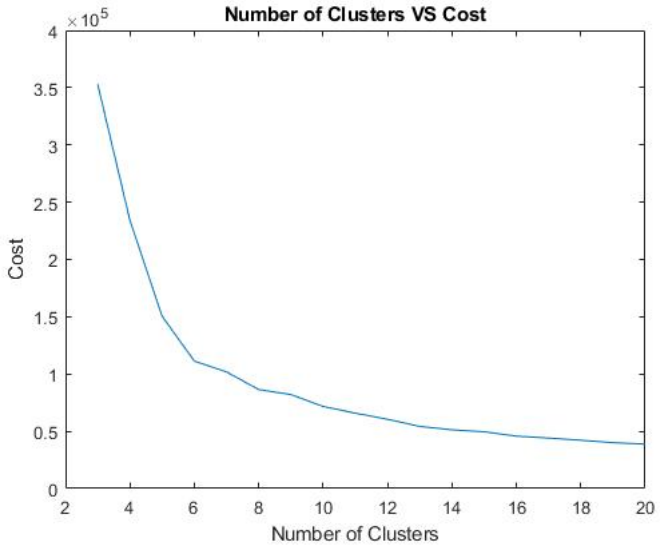

Figure 8: Relationship between the number of clusters and the cost function of k-means

In particular, the k-means algorithm is executed for different k (number of clusters) values. For each k, the algorithm is executed, using different initial representatives. The initial representatives are selected to be points of the data set that are picked in a random way, using each time a different seed. We select the minimum cost function per k value, and at the same time we store the corresponding optimal initial representatives, and we plot the relationship between the number of the clusters and the cost function (figure 8). We search for the k values that cause an important change of the slope of the line. As we can see in the figure above, there is not an important change in the slope. This is because we select the minimum cost value per number of clusters and this makes the line smoother. Also, as we have already mentioned, the dimensionality of the data set is 3 and this is related to the aforementioned behavior of the "line".

We utilize the results from the aforementioned methods and we conclude that a suitable value for the number of clusters may be equal to 6. However, we know that the number of real clusters is 8, so we will experiment with those two values.

4.1.3 K-means with 6 clusters

We execute the k-means algorithm using 6 clusters and we set the initial representatives to be the optimal initial representatives that came from the previous method. The results are shown in the figures below.

As we can see in figure 10, the algorithm predicts all the clusters adequately but cannot predict the cluster that corresponds to the "corn" cluster. In figure 12, we can notice that there are 2 clusters that are merged into one, the dark blue cluster. This is reasonable, because k-means tends to move the cluster representatives into dense regions, in order to reduce the cost function and these two clusters (that merged to one cluster) have a dense region in the upper side, so the k-means considers that this is a cluster and not two. We can claim exactly the same for the purple cluster in the same figure. So, the k-means merges some similar clusters and this is reasonable because we set the number of clusters equal to 6 and not to 8.

Based on figure 13, we can conclude that most of the clusters have a high silhouette score, which means that the data points in a cluster are similar and not similar to the data points of other clusters.

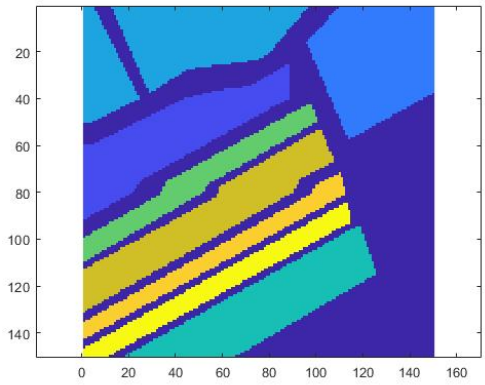

Figure 9: The actual clusters of the data set

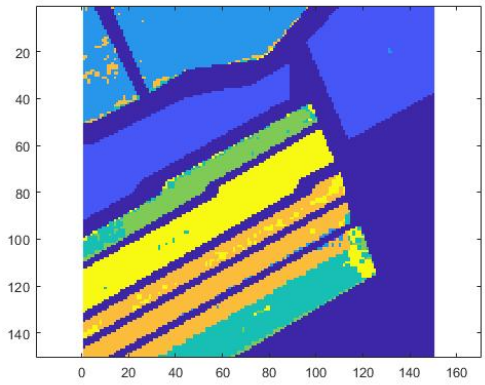

Figure 10: The predicted clusters of the data set, using k-means

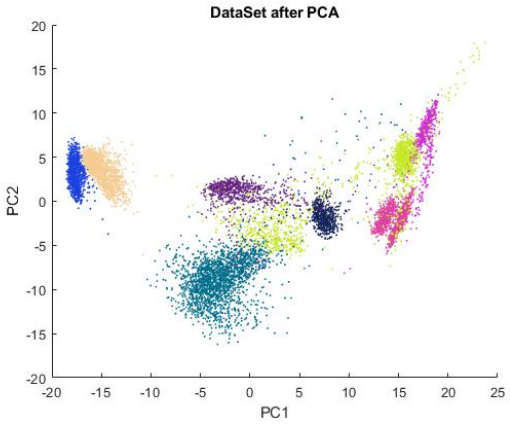

Figure 11: The actual 2D data set. Each color corresponds to a cluster

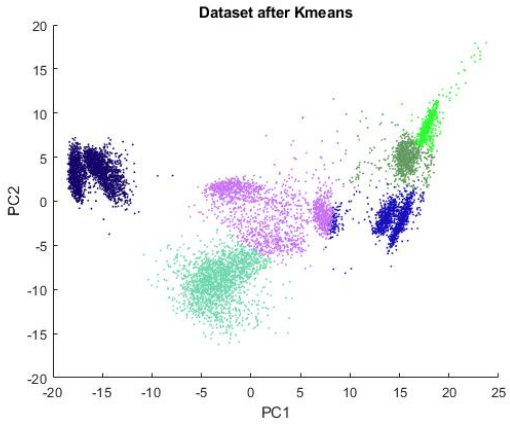

Figure 12: The clusters that come from the k-means

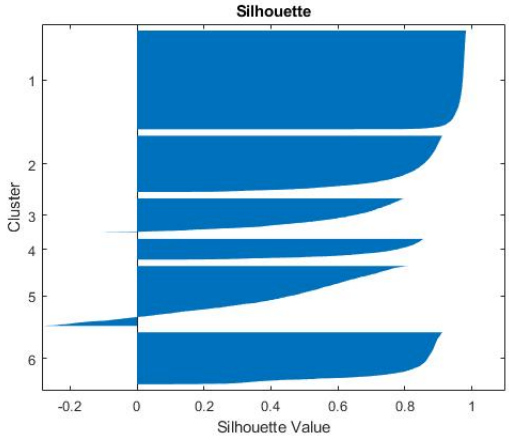

Figure 13: Silhouette score for k-means

4.1.4 K-means with 8 clusters

In the figures below, we can see the results of the k-means algorithm in the case that $k = 8$. The results are not as good as the results of the previous execution. There are many patterns that the algorithm assigned to the wrong cluster as we can see in figure 15. Some clusters, such as the "Grapes" cluster and the "Lettuce 1" are recognised by the algorithm but others are a combination of different labels. In figure 17, we notice that there are some predicted clusters that correspond to the real clusters, like the red one, but others are not predicted correctly, such as the two most left clusters that the algorithm "divided" them in the wrong way.

If we check the Silhouette plot, we conclude that the algorithm creates clusters that contain data points that are similar, but if we check the confusion matrix of the k-mean, we can understand that the algorithm does not predict the real labels as we would expect. Some clusters, such as the cluster 1, 3 and 6 correspond to the real clusters, but others, such as the cluster 2, contain data points from different real clusters. The total accuracy of the k-means is 69%, which means that the algorithm clusters a big number of pixels as we would expect, but there is enough room for improvement.

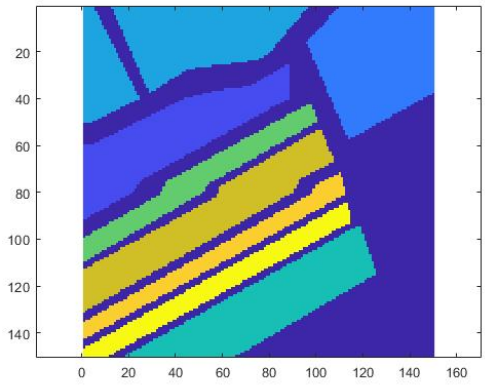

Figure 14: The actual clusters of the data set

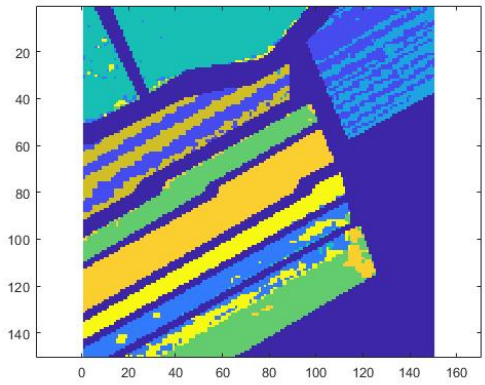

Figure 15: The predicted clusters of the data set, using k-means

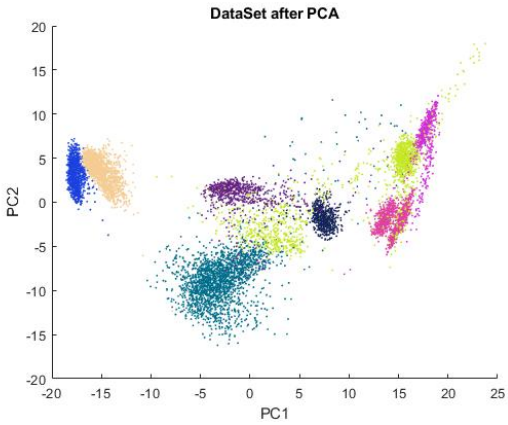

Figure 16: The actual 2D data set. Each color corresponds to a cluster

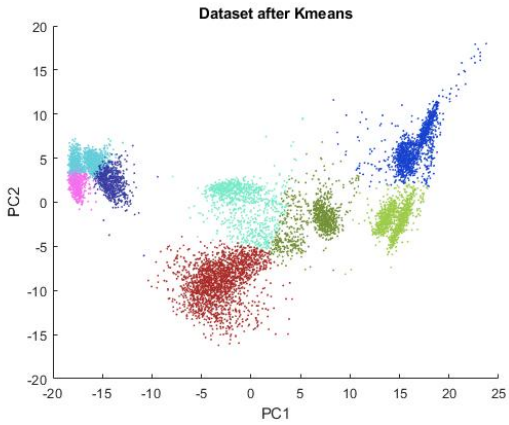

Figure 17: The clusters that come from the k-means

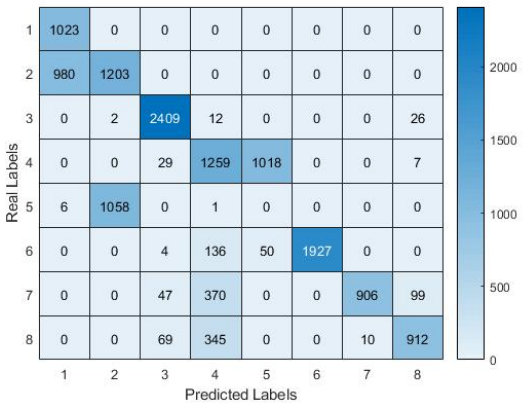

Figure 18: Confusion matrix of k-means

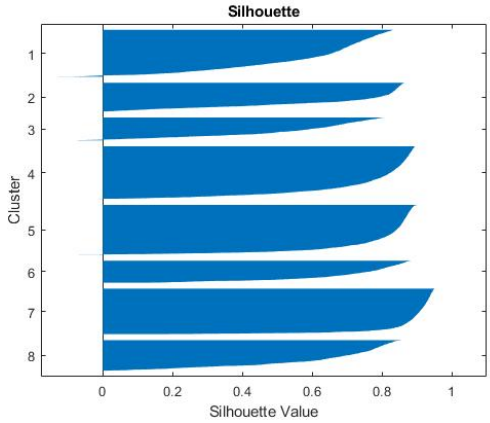

Figure 19: Silhouette score fro k-means

4.2 Fuzzy Clustering

In *fuzzy clustering* each pattern can belong to many clusters at the same time. The degree of the membership of each pattern in a cluster is related to the degree of the membership of the same pattern to all the other clusters. The degree of membership is defined by a user-defined parameter, q. If the q value is large the algorithm permits many points to belong to more than one cluster.

We are going to use point representatives, as before. So, the fuzzy clustering scheme using point representatives is called *fuzzy c-means*.

This algorithm contains two steps, that are pretty similar to the steps of the k-means. First of all, the algorithm assigns the patterns of the data set to the clusters and after that, it recalculates the representatives. These steps are repeated until a termination criterion. The termination criterion is usually related to the change of the representatives between two sequential repetitions.

It is obvious that the algorithm needs the number of clusters in order to create the appropriate representatives and finally cluster the patterns. We are going to use a method that can discover the optimal value for the parameter "number of cluster". This method is specialized for the fuzzy clustering technique.

4.2.1 Determination of number of clusters based on coefficients

This method follows the principles of the method that was described in subsection 4.1.2. In this case, we examined the relationship between the number of clusters and the corresponding cost of the k-means algorithm. However, in the case of fuzzy c-means there is additional information about the model, the fuzziness. We should take into consideration the fuzziness in order to construct a method for the prediction of a suitable number of clusters.

For this reason, we use two coefficients, instead of the cost-function value, the *partition coefficient* and the *entropy partition coefficient*. We plot the relationship between one coefficient and the number of clusters.

The partition coefficient is calculated based on the formula:

$$PC = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij}^2$$

where N is the total number of the data points, m is the number of the clusters and u_{ij} is an element of the matrix U, which is a $N \times m$ matrix that contains the degree of membership of the data point $\vec{x_i}$ in the cluster j.

The PC takes values in the range $[\frac{1}{m}, 1]$. If the value of the PC is close to 1, the clustering tends to be hard and vice versa. If the PC value is very close to the value $\frac{1}{m}$, this may indicate

that the data set do not contain physical clusters or the fuzzy c-means did not discover them.

The other coefficient that we used is the entropy partition coefficient. The calculation of PE is based on the formula:

$$PE = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij} \log u_{ij}$$

The range of the PE values is $[0, \log m]$. If the value of PE is close to the $\log m$, there is a high degree of fuzziness and this indicates, again, that the data set do not contains physical clusters or the fuzzy c-means cannot find them.

We use both of them, instead of the cost that we referred to the 4.1.2. The principles of the method are exactly the same as before and we are not going to explain them once more. In figure 20 there are the plots of the PC and the PE and the number of clusters for different q values. For $q = 1.5$ and $q = 2$ and $numberofclusters = 8$, there is an important change in the slope of the line in the PC as well as in the PE plot.This indicates that the physical clusters of the data set may be 8. We know that the actual clusters are 8, so the results of the method agree with the actual data. Furthermore, we can notice that for q values greater than 2, there are not important changes at the slope and as a result, in these cases, we cannot extract conclusions about the number of the clusters.

So, in the case of fuzzy c-means we are going to execute the algorithm only for 8 clusters, but for two different q values, $q = 1.5$ and $q = 2$.

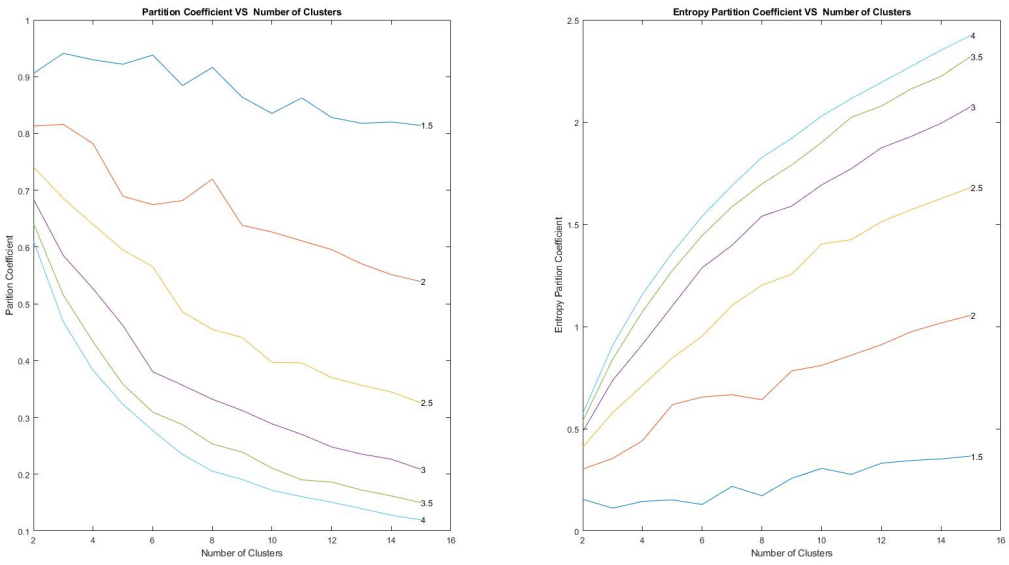

Figure 20: Partition Coefficient (left) / Entropy Partition Coefficient (right)

4.2.2 Fuzzy c-means with 8 clusters - q=1.5

The results of the fuzzy c-means with 8 clusters and $q = 1.5$ are shown below. In figure 22, there are some pixels that have the same color to the background (blue). From now on, we are going to call these pixels "background" pixels and this definition will refer to the data points that the maximum u_{ij} value is less than a user-defined threshold. These pixels are the black points of the figure 24 and they belong to more than one cluster in a great degree.

We can notice in figure 24 that the algorithm cannot recognise the two most left clusters (green-red clusters) and divide them wrongly. Also, it merges two real clusters into one (brown). These failures are reflected in the figure 22, in which some areas are well defined and some others are a kind of mixture of two different classes. The reasons for these failures have already been discussed in the k-means' section.

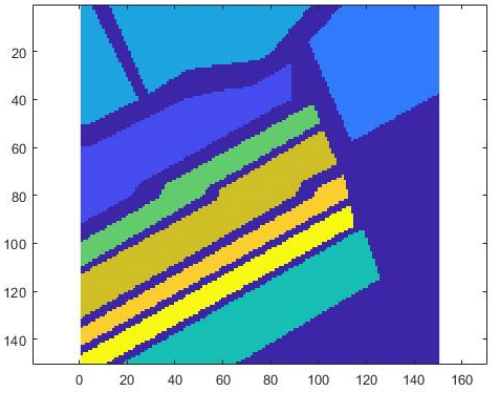

Figure 21: The actual clusters of the data set

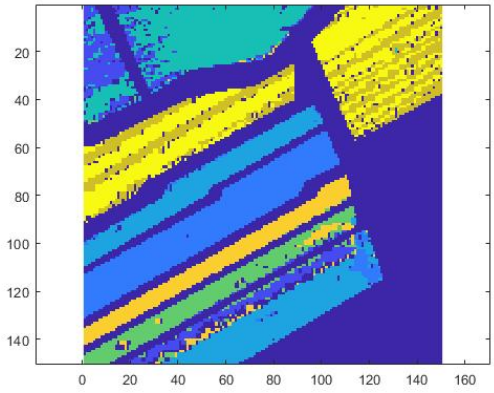

Figure 22: The predicted clusters of the data set, using fuzzy c-means (q=1.5)

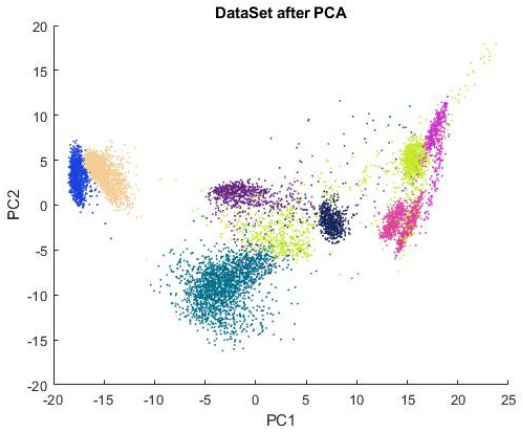

Figure 23: The actual 2D data set. Each color corresponds to a cluster

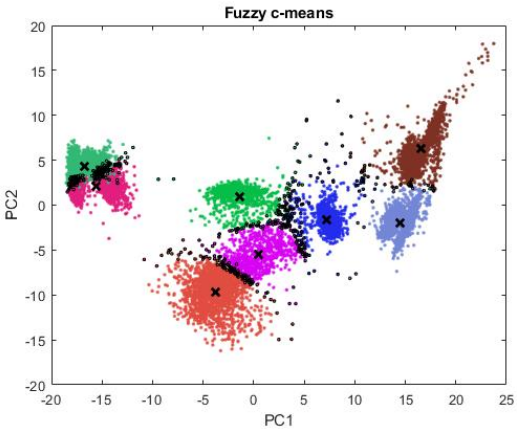

Figure 24: The clusters that come from the fuzzy c-means (q=1.5)

We execute the algorithm and we consider that each data point belongs to the cluster that correspond to the larger $u(i, j)$ value, in order to calculate the confusion matrix, the accuracy and the silhouette score for each cluster. In this point, we should mention that this approach is not 100% correct, because this algorithmic scheme is not hard, but we are going to use it in order to understand the behaviour of the algorithm.

The confusion matrix indicates that the algorithm assigns most of the patterns to the right cluster and cannot assign the patterns of the actual clusters 1 and 2 to two different clusters, so it creates a combination of these patterns as clusters. The algorithm achieves an accuracy score equal to 74%. The silhouette scores are high enough fr almost all clusters, except for cluster 6.

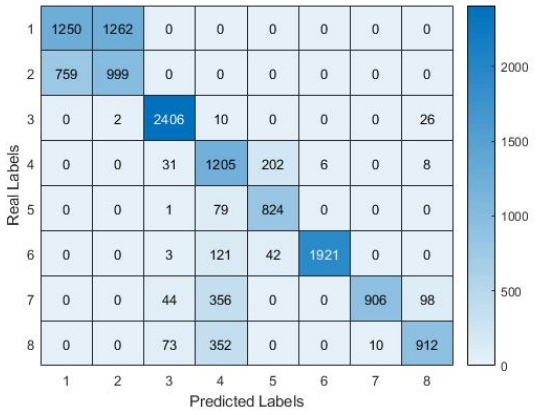

Figure 25: Confusion matrix of fuzzy c-means (q=1.5)

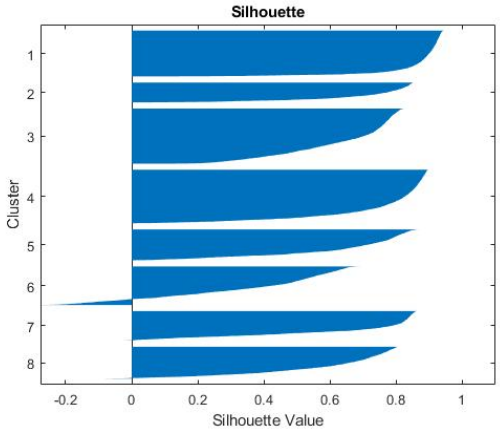

Figure 26: Silhouette score for fuzzy c-means (q=1.5)

4.2.3 Fuzzy c-means with 8 clusters - q=2

The results of the same algorithm, allowing a greater degree of membership between two or more clusters, are shown below. It is obvious that there are more "background" pixels in the crop areas than before and this is reasonable, because the q is greater than before. The clusters are more or less the same as before, but now there are more points that belong to more than one clusters in great degree. Therefore, the only difference is the points that are distributed in many clusters.

The accuracy score is smaller than before, it is equal to 69%. We consider that this small reduction is because of the increase of the q parameter. As we have already mentioned, if q is high, the algorithm permits many points to belong to more than one cluster, to a greater degree. So, this may lead to wrong final assignments of the data points to the clusters (the technique that was described in the previous section). As we can see in the confusion matrix, the predicted clusters are not as accurate as in the previous execution (with $q = 1.5$). For instance, the predicted cluster 5 does not contain any data point, which belongs to the corresponding cluster. However, the silhouette score is high for most of the clusters and that leads us to the conclusion that we do not use only one way to evaluate the clustering results.

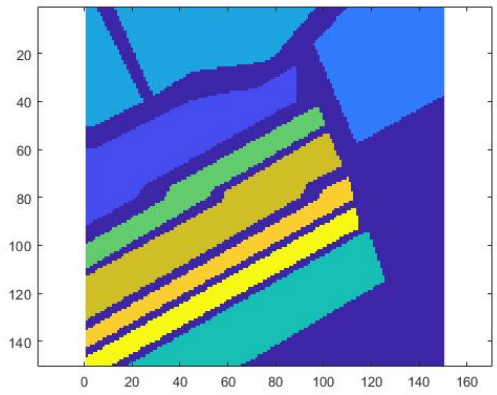

Figure 27: The actual clusters of the data set

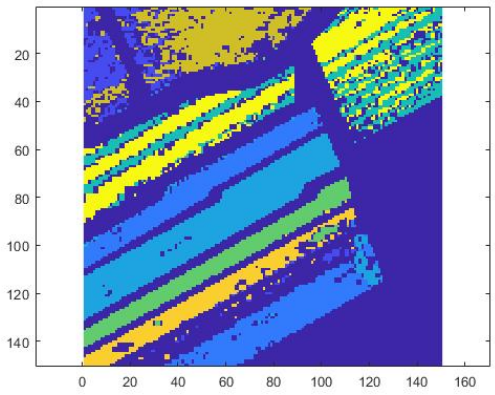

Figure 28: The predicted clusters of the data set, using fuzzy c-means (q=2)

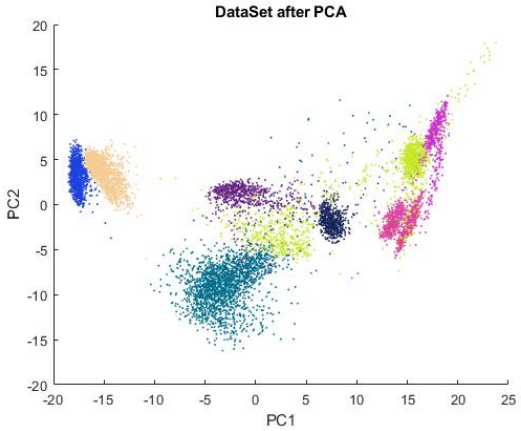

Figure 29: The actual 2D data set. Each color corresponds to a cluster

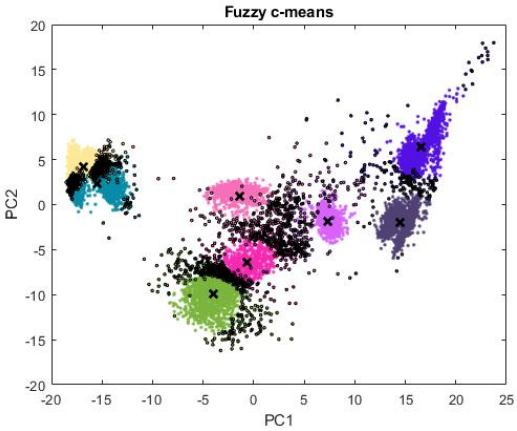

Figure 30: The clusters that come from the fuzzy c-means (q=2)

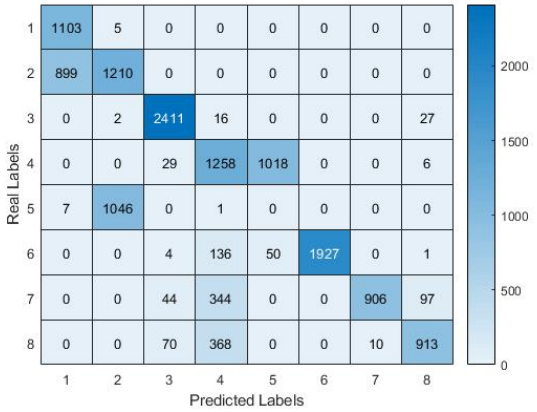

Figure 31: Confusion matrix of fuzzy c-means (q=2)

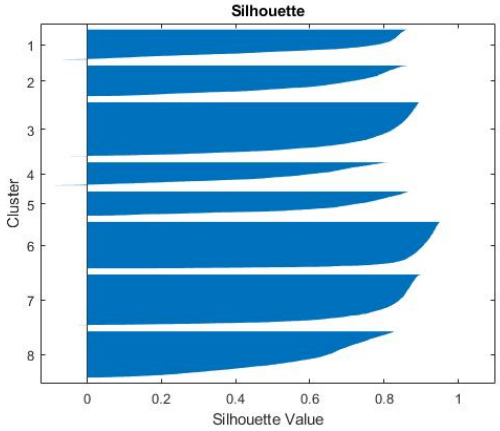

Figure 32: Silhouette score for fuzzy c-means (q=2)

4.3 Possibilistic Clustering

The *possibilistic clustering* belongs to the CFO family of algorithms. In this case, each pattern has a degree of compatibility with the clusters. One may think that fuzzy clustering is similar to possibilistic clustering, but this is a wrong conclusion, because in possibilistic clustering a pattern can belong to one cluster, regardless of the fact that it will probably belong to other clusters.

An advantage of the possibilistic clustering algorithm is that it does not need the number of clusters in order to create clusters, because this algorithm searches for dense regions of data points and moves the cluster representatives to those regions. In this case, we use points as cluster representatives. On the other hand, the k-means and the fuzzy c-means are partitioning algorithms and as a result they need the number of clusters in order to produce a clustering.

We should emphasize that the algorithm takes as a parameter an $\vec{\eta}$ vector. This parameter defines the power that a pattern has in computing the representative of a cluster. For finding the appropriate $\vec{\eta}$ vector we can use the results from the fuzzy c-means. For the calculation of the j-th component of the $\vec{\eta}$ vector, we used the formula:

$$\eta_j = \frac{\sum_{i=1}^{N} u_{ij}^q d(\vec{x_i}, \vec{\theta_j})}{\sum_{i=1}^{N} u_{ij}^q}$$

We should emphasize that the q has a different meaning now. A large value means that the patterns are distributed to all clusters to a great degree and vice versa.

We execute the possibilistic k-means using a large number of clusters in order to deploy the advantage that has already been discussed. The results are shown in figure 35. We notice that the cluster representatives are around the regions of the mean vectors of the actual

representatives. Therefore, the algorithm discovers the regions that correspond to the real clusters and this is reasonable, because the actual clusters are compact and as we have already discussed the algorithm can discover dense regions and set them as clusters.

We executed the algorithm, using different values for the q parameter, different number of clusters and as a result different $\vec{\eta}$ vectors. Some of the algorithm results, using different combinations of parameters are shown below. In this point, we should emphasize that this kind of visualization is not 100% correct, because in the possibilistic clustering each point belongs to many clusters.

First of all, we noticed that the smallest the q value is, the better the results, so we omit the visualizations for large q values. Also, we noticed that for small values of k, the algorithm creates bigger clusters, that include the physical clusters, than the real clusters. However, for very large k values the algorithm creates many clusters and we cannot make safe conclusions about the physical clusters.

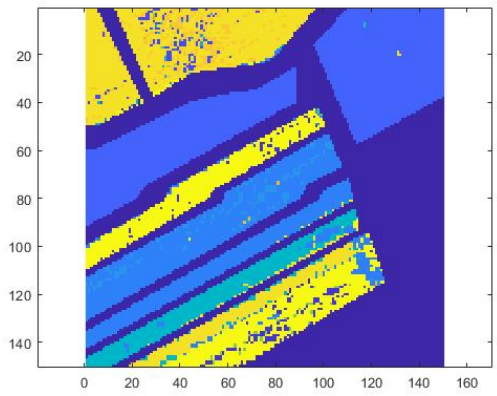

Figure 33: The predicted clusters of the data set, using possibilistic k-means (q=1.5, k=16)

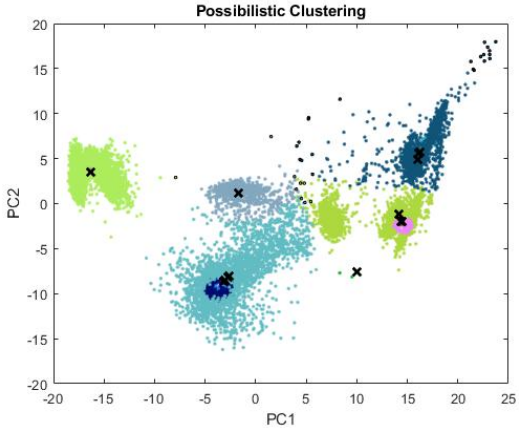

Figure 34: The clusters that come from the possibilistic k-means, in 2D space (q=1.5, k=16)

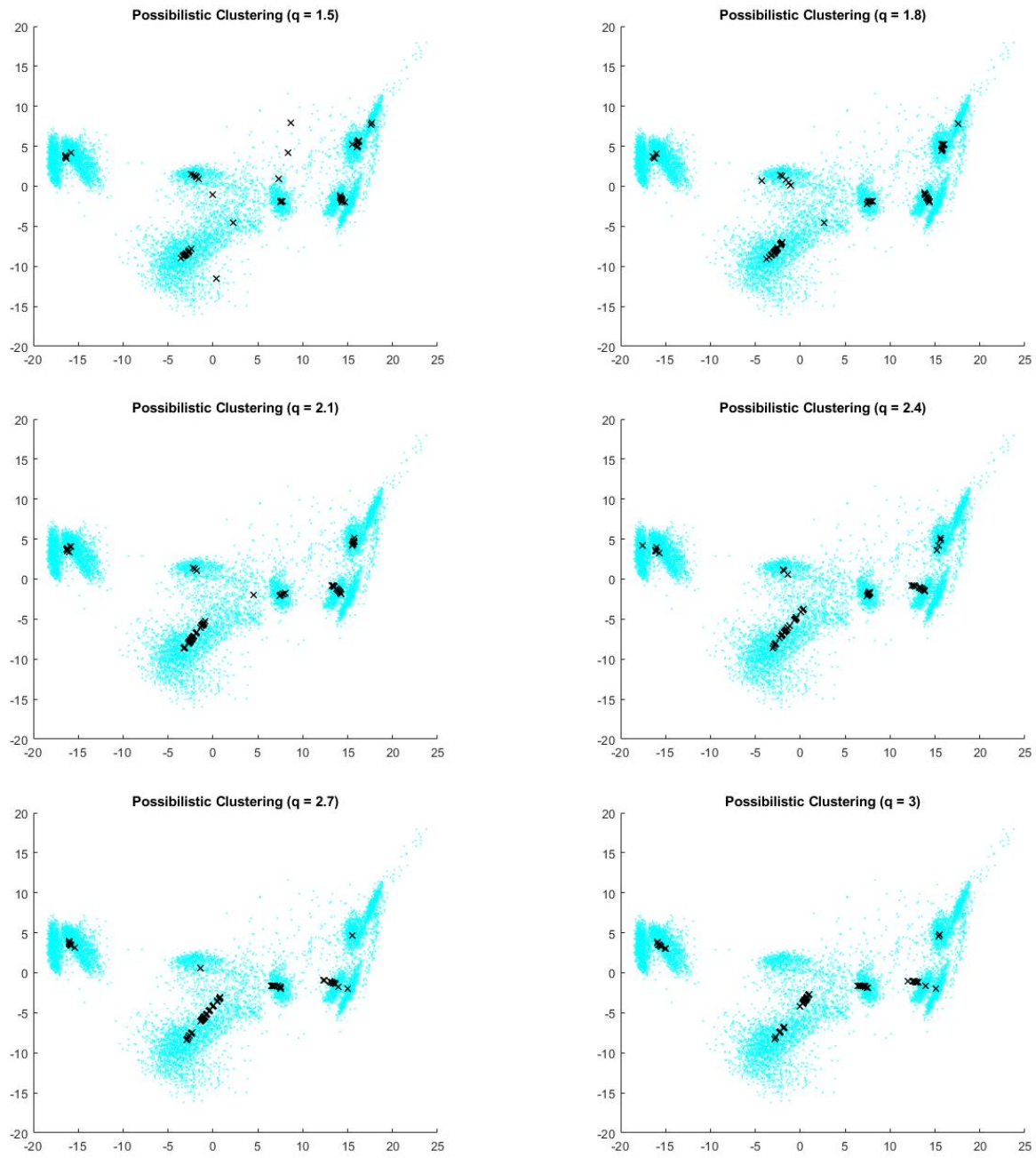

Figure 35: Results of possibilistic clustering with 100 points as cluster representatives and different q values

4.3.1 Possibilistic k-means with 6 clusters

We execute the possibilistic k-means and set the number of the clusters equal to 6. We use as q the values 1.5 and 2.5 and we do not use higher values, because after many experiments we concluded that the clustering results are bad, using high q values. The results of the executions are below.
Small changes at the q value may lead to different clustering results. However, most of the predicted clusters do not correspond to the actual clusters and this is reflected in the valley figures. Only the cluster "Lettuce 3" is predicted by the algorithm, for both q values. In general terms, the algorithm mix patterns that have different labels in the same cluster.

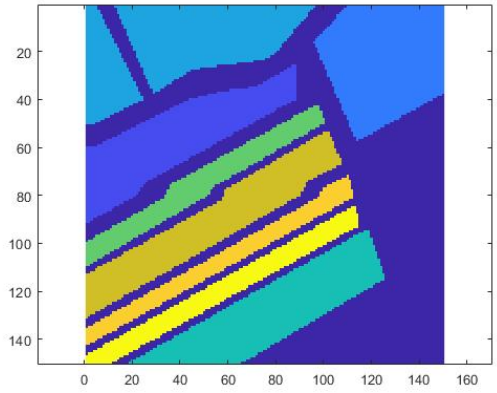

Figure 36: The actual clusters of the data set

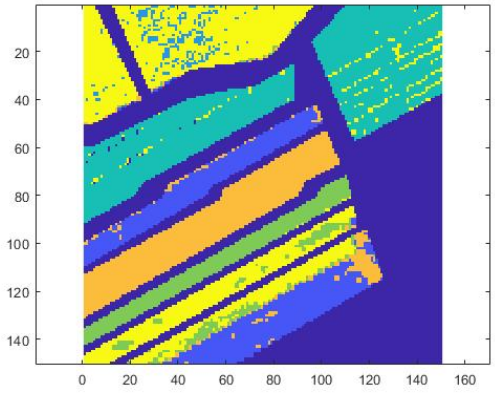

Figure 37: The predicted clusters of the data set, using possibilistic k-means(q=1.5)

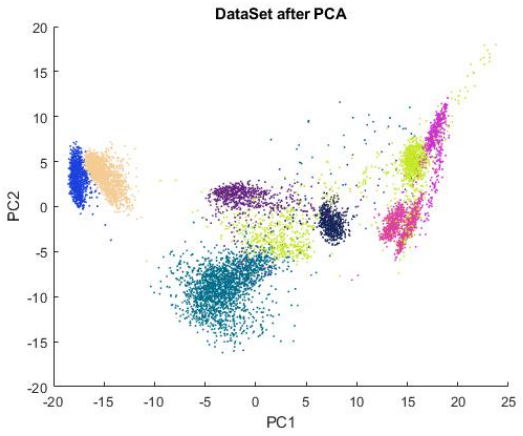

Figure 38: The actual 2D data set. Each color corresponds to a cluster

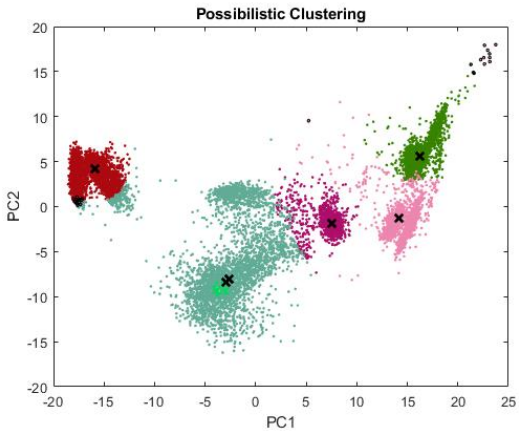

Figure 39: The clusters that come from the possibilistic k-means (q=1.5)

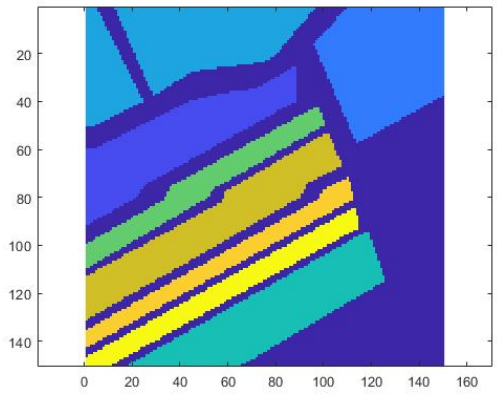

Figure 40: The actual clusters of the data set

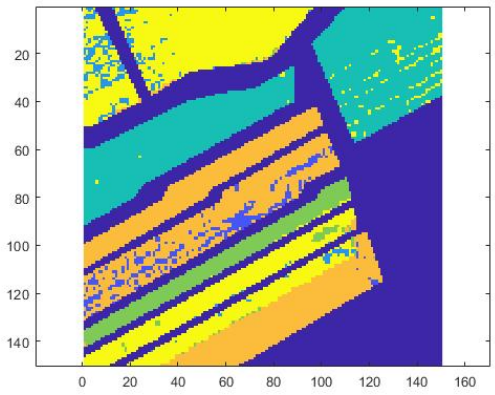

Figure 41: The predicted clusters of the data set, using possibilistic k-means(q=2.5)

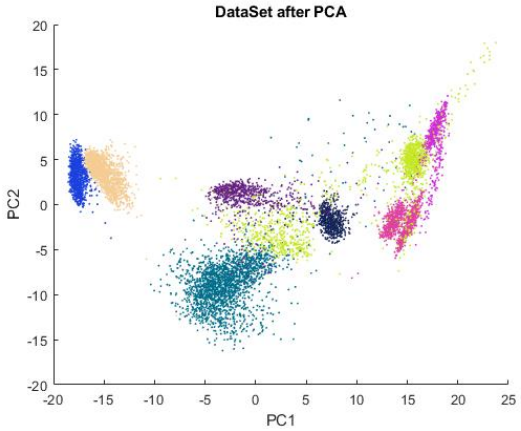

Figure 42: The actual 2D data set. Each color corresponds to a cluster

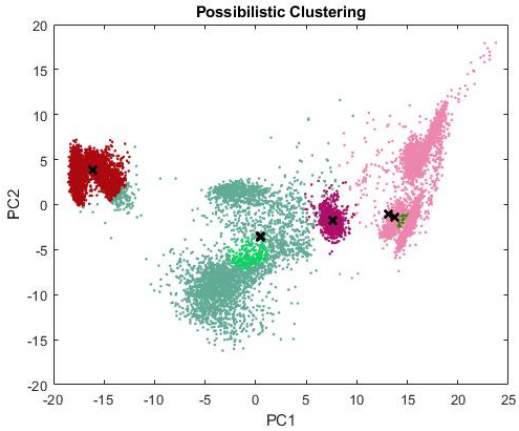

Figure 43: The clusters that come from the possibilistic k-means (q=2.5)

As we can see in figures 44 and 45, the silhouette scores are not high for all the clusters and this result is shown for the first time up to now. The possibilistic k-means did not achieve to create clusters that all of them contain patterns with high similarities.

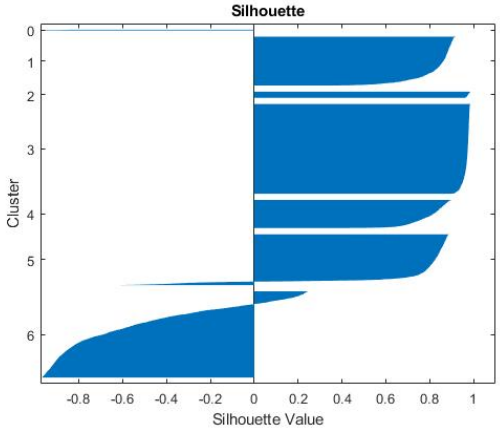

Figure 44: Silhouette score for possibilistic k-means (q=1.5)

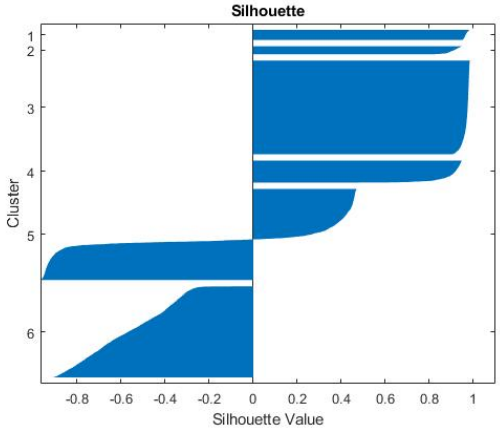

Figure 45: Silhouette score for possibilistic k-means (q=2.5)

4.3.2 Possibilistic k-means with 8 clusters

We executed the possibilistic k-means, using different q values. A q value that creates as accurate clusters as possible is $q = 1.7$.

Generally, the possibilistic k-means predicts some clusters (for example in figure 47, the cluster "grapes") or some bigger clusters that contain more than one relative clusters (for example, there is a cluster that contains the two sub-clusters: broccoli 1, broccoli 2). Other clusters are a mix of different clusters.

If we check the confusion matrix, we will see that the predicted cluster 1 does not contain any data point. Also, in figure 49, we notice that the representatives do not reach all the dense areas and as a result the clustering is not as we would expect. This is reasonable, because the representatives try to find dense areas and because in this case some areas contain more data points than others, the representatives are magnetized from these areas to a greater degree. The algorithm achieved 68% total accuracy.

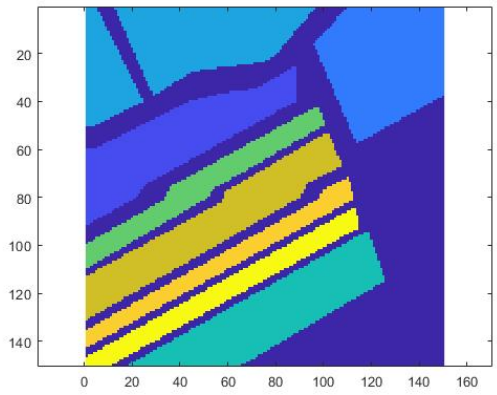

Figure 46: The actual clusters of the data set

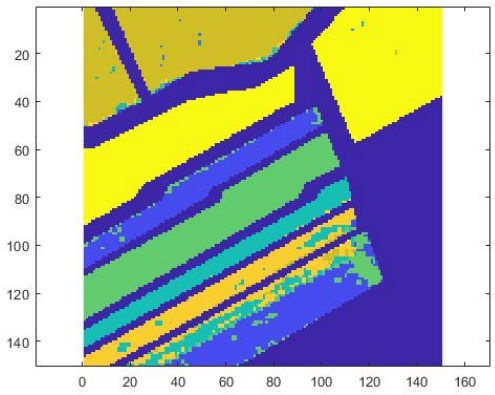

Figure 47: The predicted clusters of the data set, using possibilistic c-means (q=1.7)

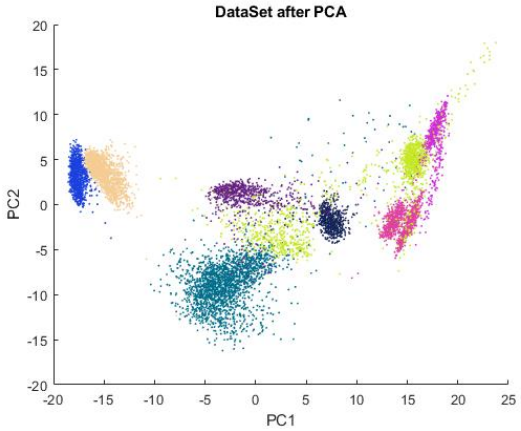

Figure 48: The actual 2D data set. Each color corresponds to a cluster

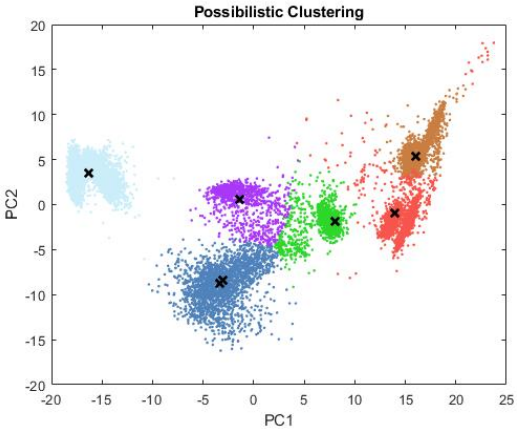

Figure 49: The clusters that come from the possibilistic c-means (q=1.7)

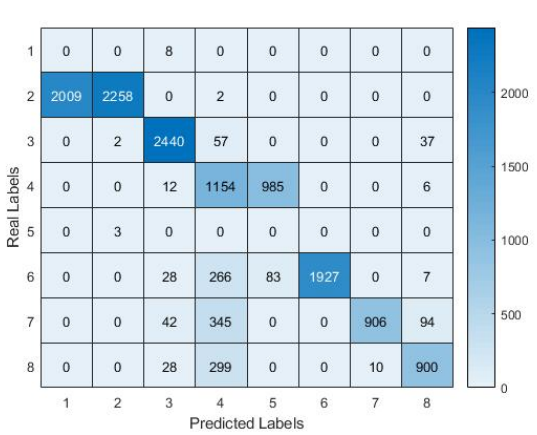

Figure 50: Confusion matrix of possibilistic c-means (q=1.7)

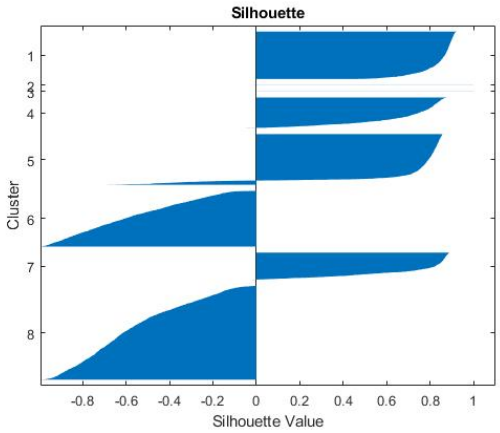

Figure 51: Silhouette score for possibilistic c-means (q=1.7)

4.4 Probabilistic Clustering

In this kind of clustering, each data point, $\vec{x_i}$, belongs to a cluster C_j with $P(C_j|\vec{x_i})$, where $i = 1, ..., N$ and $j = 1, ..., m$. The data point is assigned to the cluster with the largest probability. The whole concept is based on the Bayes theory. However, we do not have a training data set, which means a data set that contains the classes of the data points, in order to use it for the appropriate calculations that are required in order to find the values of the aforementioned conditional probabilities.

We suppose that the data set contains clusters and the data points of each cluster come from a Gaussian distribution. Each Gaussian distribution is characterised by a *mean vector* and a *covariance matrix*. If we know these two parameters, we can reproduce the data points of the cluster. However, as we have already mentioned, we do not have a train data set, so the goal is to *estimate* these parameters. For the estimation, we use a well-known algorithm, which is called *Expectation Maximization* (EM).

The EM algorithm is an iterative algorithm. It contains two steps. In the first step, the algorithm estimates the mean vector, the covariance matrix, as well as, the a priori probabilities of the clusters. This step is called *expectation step*. In the second step, the *maximization step*, the algorithm calculates the conditional probabilities $P(C_j|\vec{x_i})$. These two steps are repeated until the sum of the absolute differences of the a priori probabilities, the mean vectors and the covariance matrices between two iterations is smaller than a user-defined threshold.

We executed this algorithm, using each time a different set of parameters. In particular, we experiment with the maximum number of iterations, the termination threshold and the number of clusters.

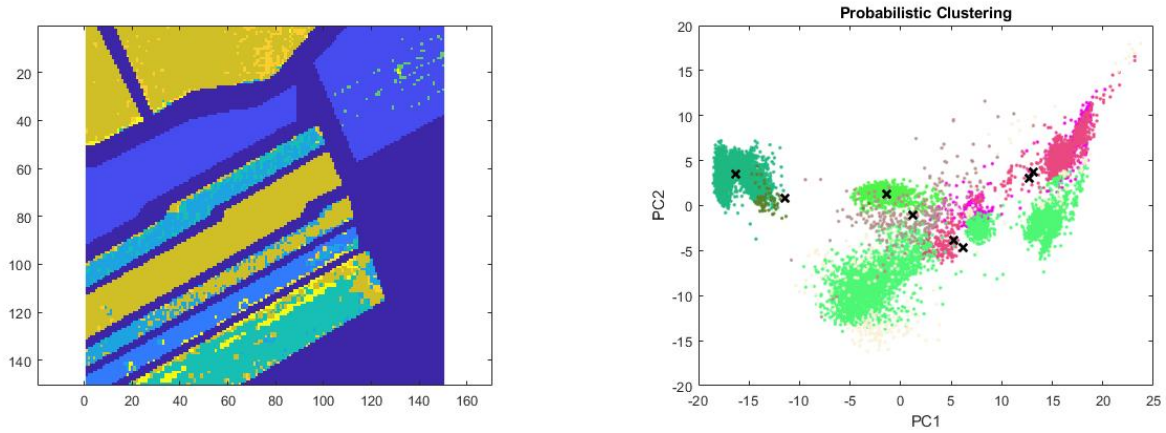

Figure 52: The clusters that come from the probabilistic approach (20 iterations of the EM algorithm)

If the algorithm does few iterations, as we can see in figure 52, it has not the appropriate time, so as to estimate all the cluster representatives and as a consequence, the clustering is not as accurate as we expected.

If we permit more iterations, we get better results, because the algorithm can estimate the mean vectors, the covariance matrices and the a priori probabilities more accurately. If the termination threshold is equal to 0.001 and the maximum number of iterations is 1000, the EM gives the results of the figure 53. The algorithm cannot identify the two most left clusters (like most of the aforementioned algorithms), neither the three clusters in the right side. Also, it creates a new cluster, in the "center" of the 2D space, the dark blue cluster, which contains few data points.

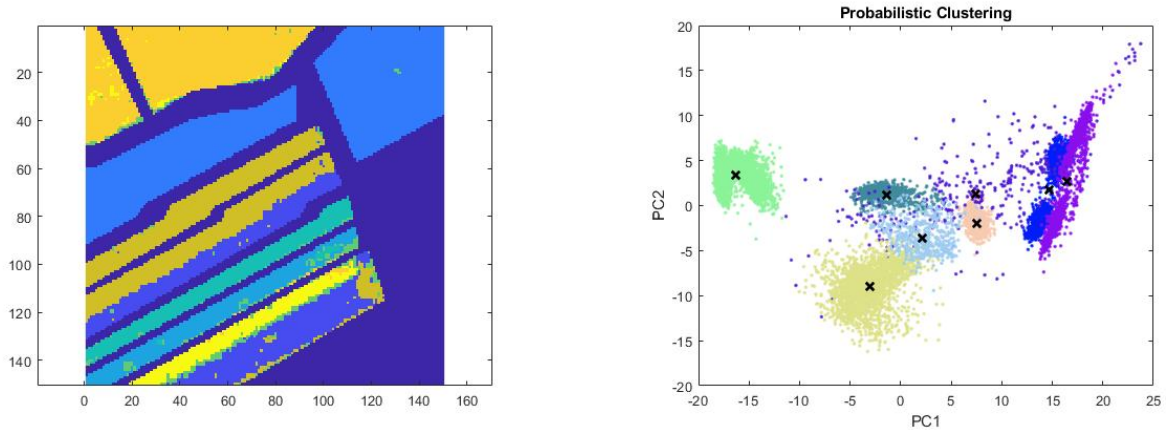

Figure 53: The clusters that come from the probabilistic approach (1000 iterations of the EM algorithm and termination threshold = 0.001)

If we reduce the termination threshold and let the maximum number of iterations as is, the EM algorithm creates a better clustering. In particular, if the termination threshold is equal

to 0.000001 the algorithm gives the below clustering. It achieved to discover the two clusters on the left side, but it did not discover the three clusters on the right side.

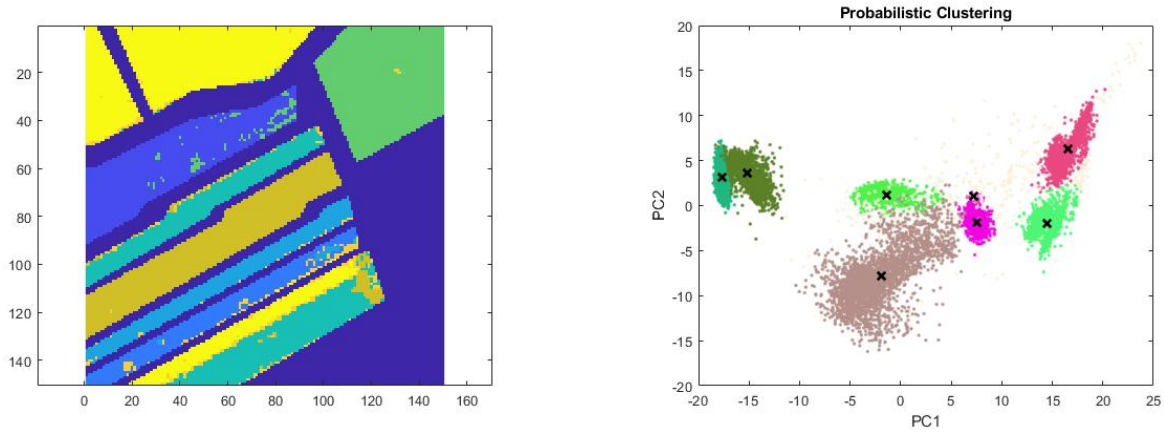

Figure 54: The clusters that come from the probabilistic approach (1000 iterations of the EM algorithm and termination threshold = 0.000001)

Also, we execute the EM algorithm using a different number of clusters, in order to check if the algorithm can discover all the physical clusters. We give the results for 9 and 10 clusters. We can conclude that the results are not improved to a great degree.

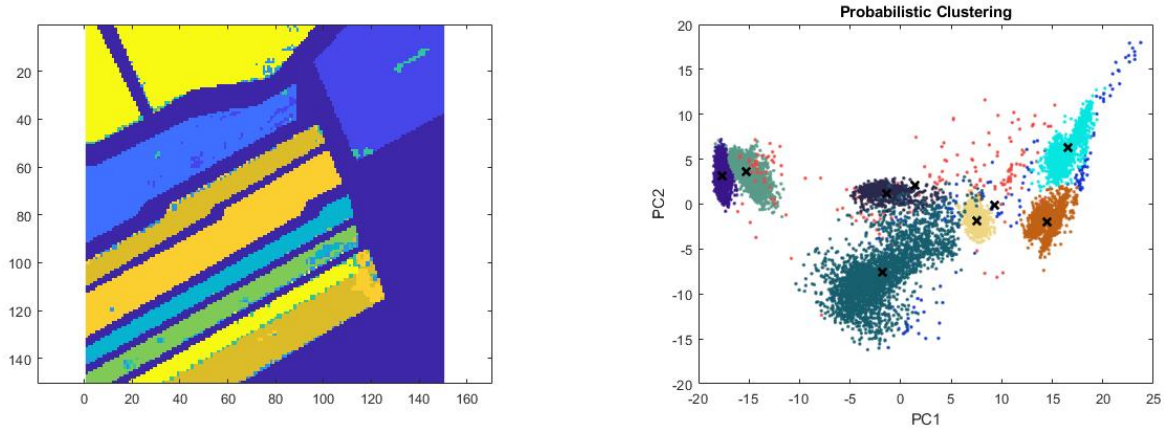

Figure 55: The 9 clusters that come from the probabilistic approach

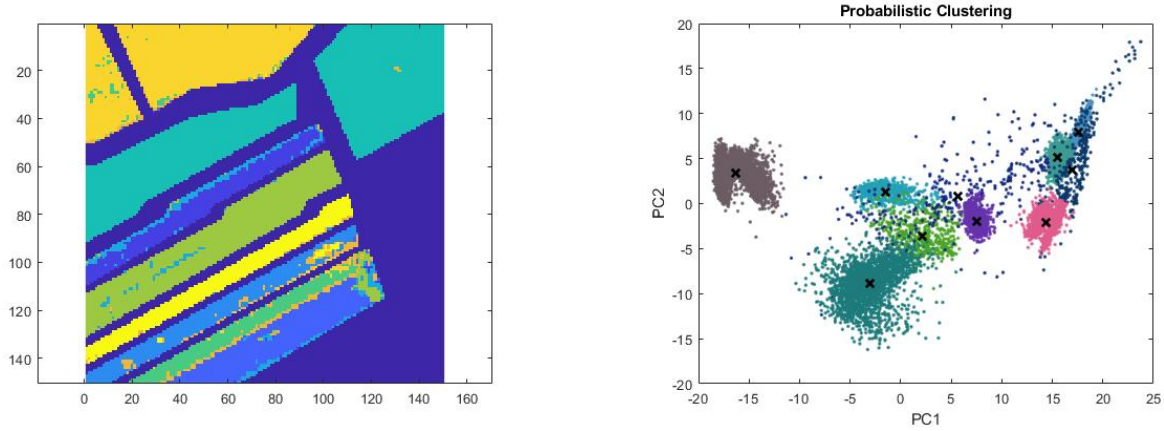

Figure 56: The 10 clusters that come from the probabilistic approach

We should emphasize that the EM algorithm needs more time in order to produce a clustering, compared to the previous algorithms, but we get, more or less, more accurate results than the previous ones. So, we should decide if we prefer more accurate clustering or less execution time in order to decide if the EM is suitable for this problem.

Also, we should emphasize that the algorithm had never reached the maximum number of iterations.

After these experiments, we execute the algorithm for 6 and 8 clusters, in order to have the chance to compare the results to the previous results.

4.4.1 Probabilistic k-means with 6 clusters

We execute the algorithm and we set the number of clusters equal to 6. The results are below.

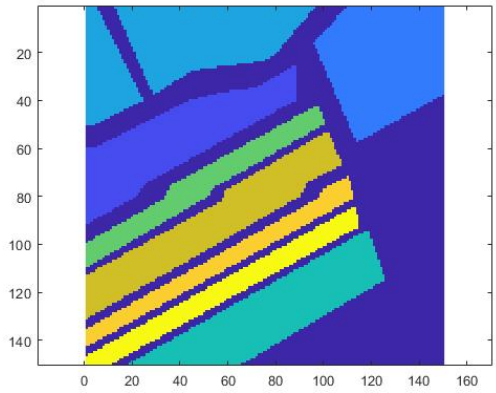

Figure 57: The actual clusters of the data set

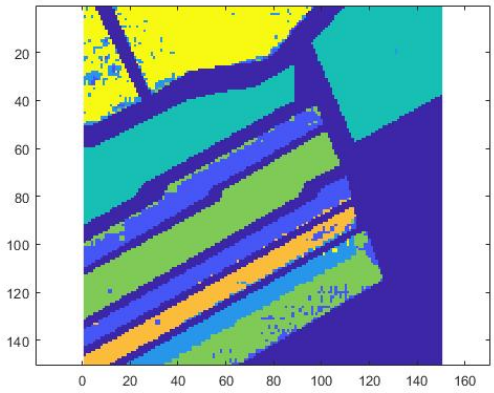

Figure 58: The predicted clusters of the data set, using probabilistic k-means

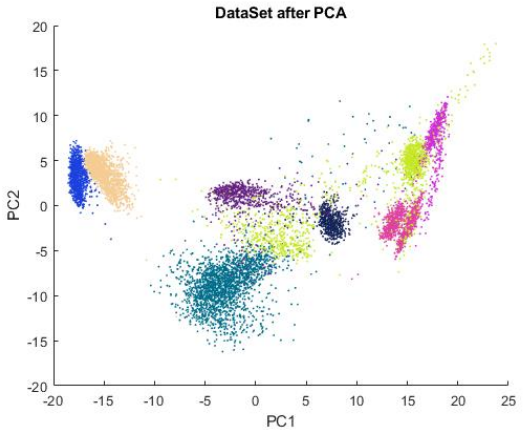

Figure 59: The actual 2D data set. Each color corresponds to a cluster

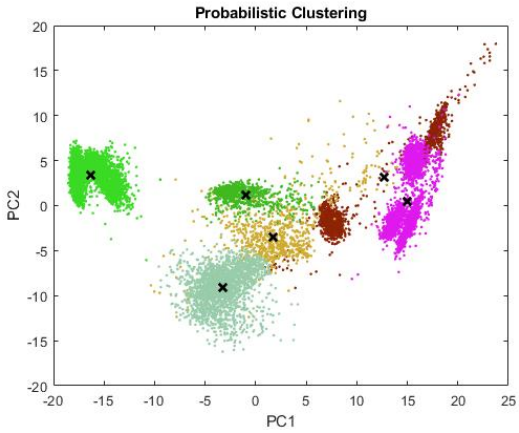

Figure 60: The clusters that come from the probabilistic k-means

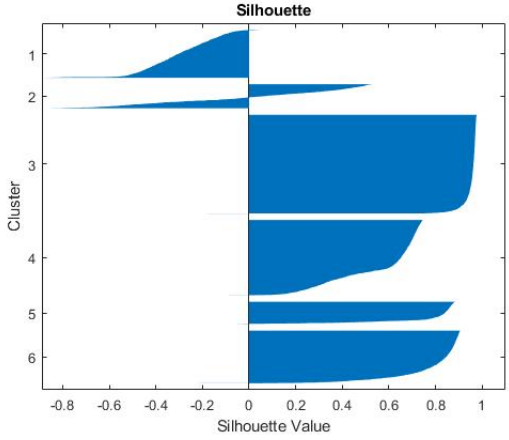

Figure 61: Silhouette score for probabilistic k-means

The algorithm does not achieve to divide the two "broccoli" clusters, as usual, can recognize the "grapes" cluster and the "Lettuce 4" cluster. In general, most of the representatives are located in the right area, but the assignment of the patterns to the clusters is not accurate. Finally, the silhouette score is very high for most of the clusters, but there are exceptions.

4.4.2 Probabilistic k-means with 8 clusters

Also, we execute the probabilistic k-means with 8 as number of clusters.

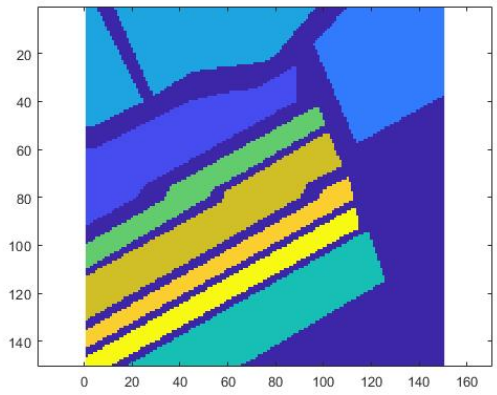

Figure 62: The actual clusters of the data set

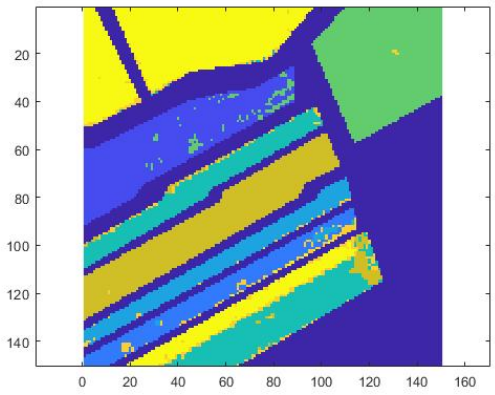

Figure 63: The predicted clusters of the data set, using probabilistic k-means

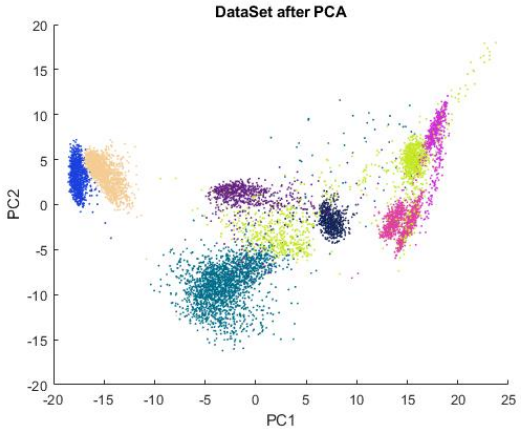

Figure 64: The actual 2D data set. Each color corresponds to a cluster

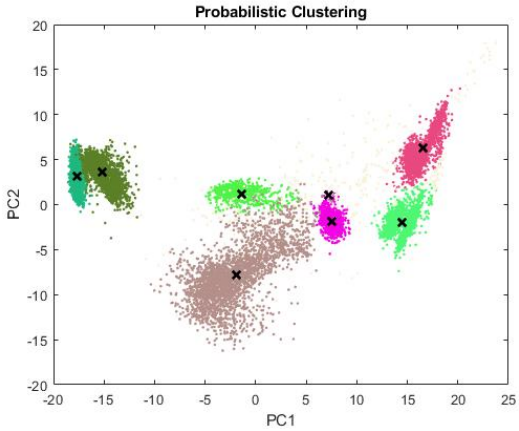

Figure 65: The clusters that come from the probabilistic k-means

The results are better, compared to all the previous results, but not 100% accurate. The algorithm recognizes the clusters "broccoli 1" and "broccoli 2" as two different clusters. Also, it recognizes the clusters "grapes", "lettuce 2" and "lettuce 4", but the other clusters are a mixture of two or more different labels. Of course, the aforementioned results are reflected in figure 63, as well. It is obvious that the clusters are closer to the real clusters and this is ascertained by the confusion matrix, in which the majority of the patterns are in the diagonal, as well as, by the total accuracy, which is equal to 83%, which is the best accuracy score, up to now.

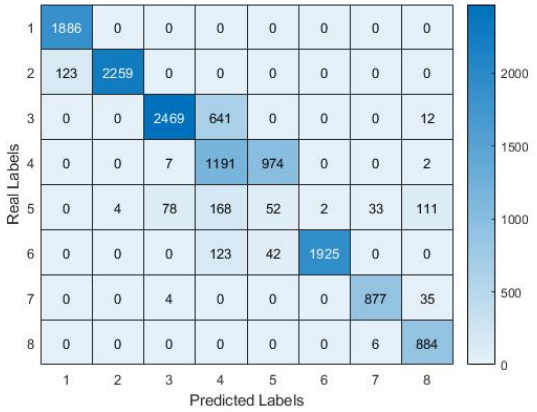

Figure 66: Confusion matrix of probabilistic k-means

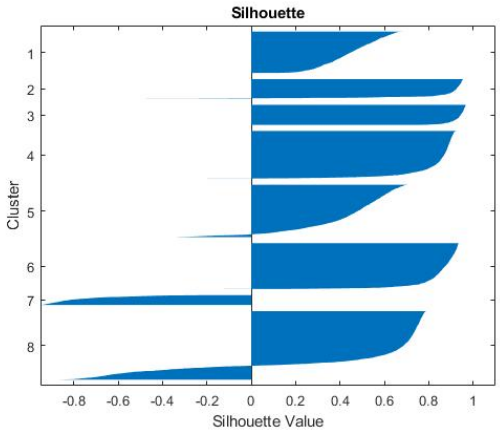

Figure 67: Silhouette score for probabilistic k-means

5 Hierarchical Algorithms

Another category of clustering algorithms is the *hierarchical algorithms*. These algorithms produce a hierarchy of many different clusterings for a data set. There are two approaches, the *agglomerative* approach, in which in the first step of the algorithm, every point is considered to be a cluster and after each step two clusters are merged. This procedure stops when all the data points belong to the same cluster. The second approach is the *divisive hierarchical clustering algorithms* that follow the opposite direction. We are going to use agglomerative algorithms in this study.

In the agglomerative approach, in each iteration of an algorithm the closest clusters are selected in order to be merged and to construct a bigger cluster. The distance between two clusters is a parameter, which creates different agglomerative approaches. In particular, the crucial point is the calculation of the distance between a merged cluster C_q, which contains the clusters C_i and C_j and another cluster C_s. We use three different approaches to this calculation.

5.1 Complete-Link algorithm

As we have already mentioned, in each step of an agglomerative algorithm two clusters, C_i and C_j, are merged and create a new cluster, C_q. An important aspect of an agglomerative clustering algorithm is the way that it computes the distance between the cluster C_q and another cluster C_s.

In the *Complete-Link algorithm* (CL) this distance, d, is computed based on the formula:

$$d(C_q, C_s) = max\{d(C_i, C_s), d(C_j, C_s)\}$$

The algorithm computes these distances for each C_s cluster that the current clustering contains and selects the smallest one. This means that the algorithm tries to create small compact clusters from the start of the execution. These clusters are merged as the execution of the algorithm continues. Therefore, the algorithm has two important features:

1. It tends to form compact clusters.
2. It forms the clusters at high dissimilarities.

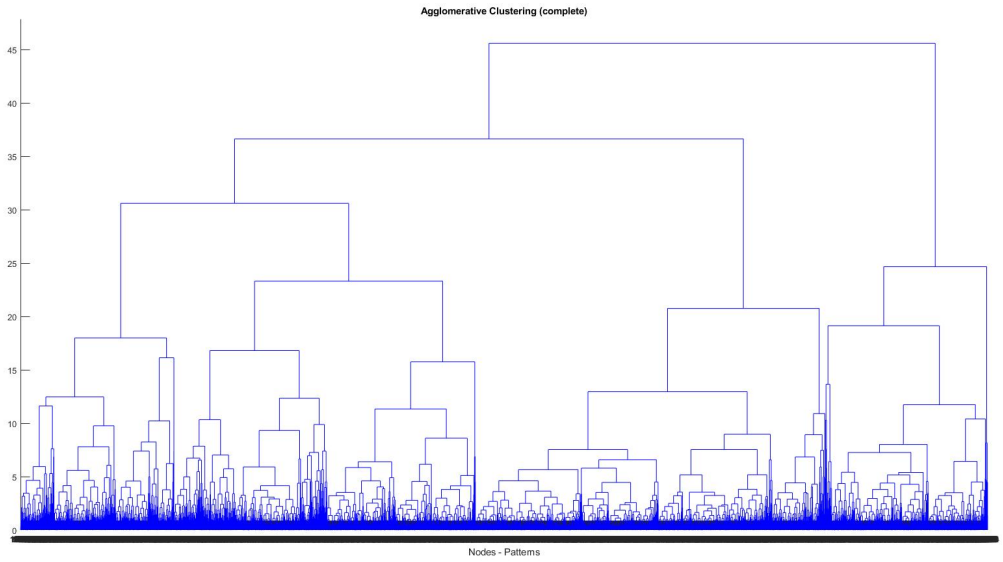

Figure 68: Dendrogram, using complete-link algorithm

We use the complete-link algorithm in this study, because the clusters are compact enough. First of all, we create the dendrogram of the clusterings, so as to check in which way the algorithm constructs the clusterings and after that we execute the algorithm, using different numbers as clusters.

From the dendrogram in figure 68, we conclude that the algorithm creates many "small" clusters in the beginning and merges them as the execution goes one. Also, we can notice that there is a cluster (the cluster that corresponds to the most right part of the dendrogram), that contains few data points. Although these data points belong to a specific cluster, in reality, the algorithm cannot merge them with another cluster. This cluster is the highest cluster (dark blue cluster) of the fourth plot of the figure 69. Based on the dendrogram, we can conclude that the data set contain 6 physical clusters.

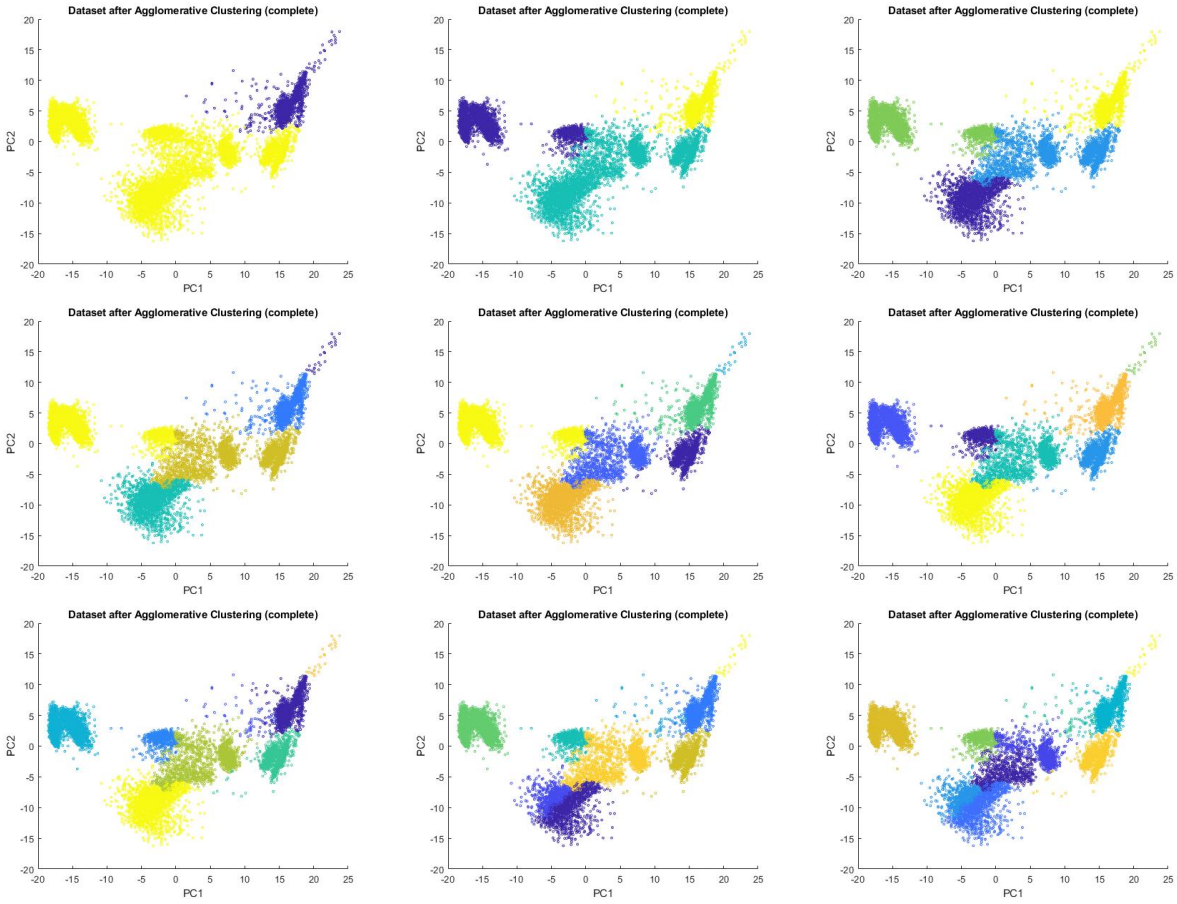

Figure 69: Clusterings that are produced using complete linkage. Each clustering contains a different number of clusters

If we check the figure 69, we will conclude that the algorithm creates compact clusters. Also, we understand that the algorithm tries to create clusters that have approximately the same diameters. So, in each step it usually divides the cluster with the bigger diameter. In this case, this feature is not desirable, because we know that the clusters do not cover the same area, so they do not have the same diameter. This leads to wrong assignments of data points to the clusters.

In this point, we should emphasize that the algorithm merges small clusters into bigger clusters, but we use the term "divide" in order to present the different clusterings. We consider that is much easier for the reader to study the clusterings that the algorithm produces with the inverse strategy. That's why we present first the clustering that contains two clusters and last the clustering that contains 10 clusters.

5.1.1 Complete-link with 6 clusters

We execute the clustering algorithm using 6 clusters. The results are shown below.

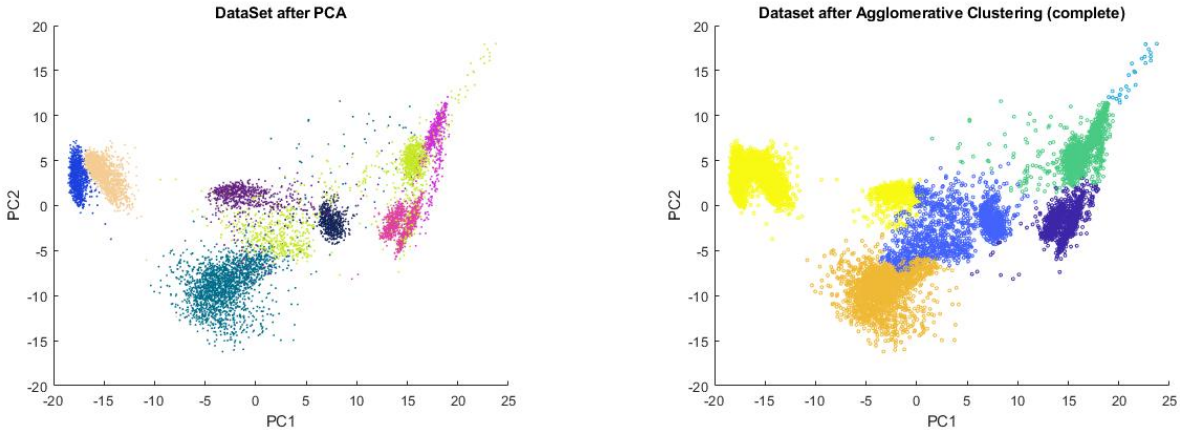

Figure 70: Complete-link with 6 clusters (left: the real clusters / right: the predicted clusters)

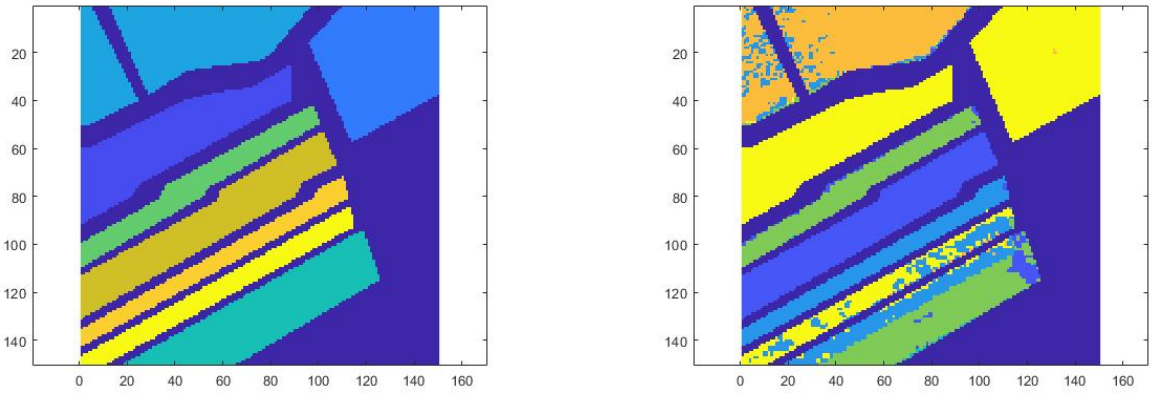

Figure 71: Complete-link with 6 clusters - valley image (left: the real clusters / right: the predicted clusters)

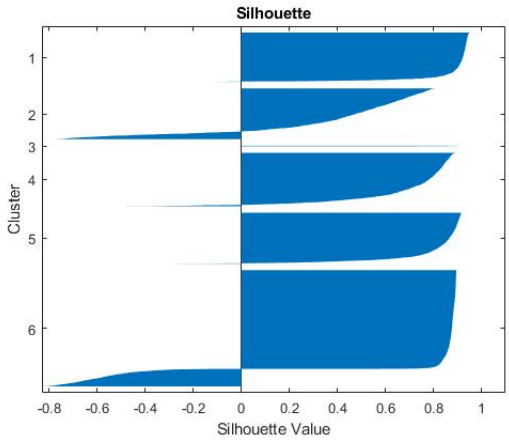

Figure 72: Silhouette score for complete-link

As we can see in figure 70, the algorithm cannot recognize all the clusters and some of them are merged in a larger cluster, like the yellow one or the green one. If we check the figure 71, we can conclude that the algorithm recognize perfect some crops, like the blue one or the green one, but others are merged into one crop like the yellow one and some crops contain pixels from different clusters, so they are a combination of different vegetables.

If we check the silhouette plot, we will understand that the clusters contain data points that are not similar to one another.

5.1.2 Complete-link with 8 clusters

Also, we execute the algorithm using 8 clusters, because the data set contains 8 different crops. We notice that the algorithm divides some clusters from the previous clustering (the clustering with the 6 clusters), but the resulting clustering is not the same as the real clustering. Some of the predicted clusters are approximately the same as the real ones, like the yellow one in the figure 73, but others are still a union of smaller clusters, like the azure one in the same figure.

If we compare the valley figure after this execution with the previous valley figure, we will notice that some crops contain more pixels that do not belong to this crop, like the blue one in the figure 74. Furthermore, if we compare the matrices of this execution and the previous one, we will understand that the algorithm divides the clusters into smaller clusters, because in this execution we set as number of clusters a greater number, but this division cannot produce the actual clusters.

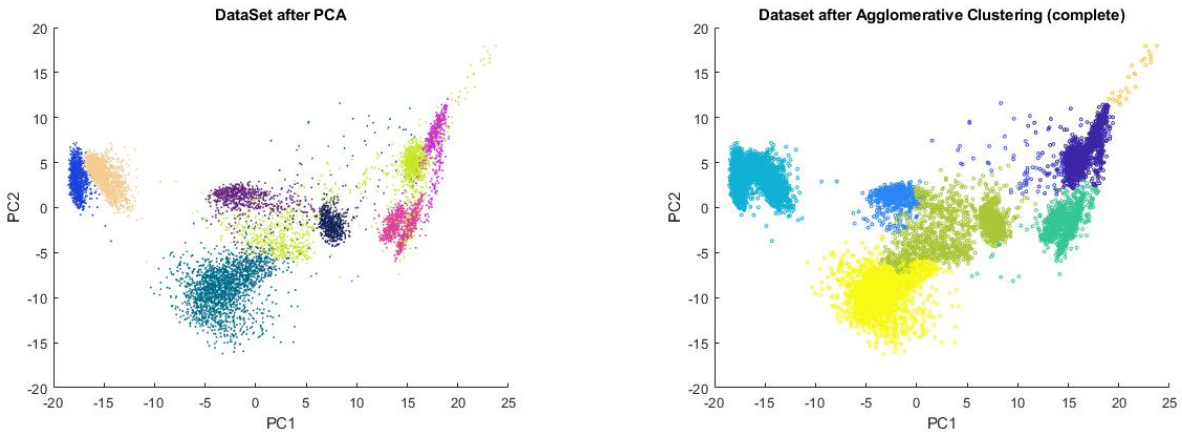

Figure 73: Complete-link with 8 clusters (left: the real clusters / right: the predicted clusters)

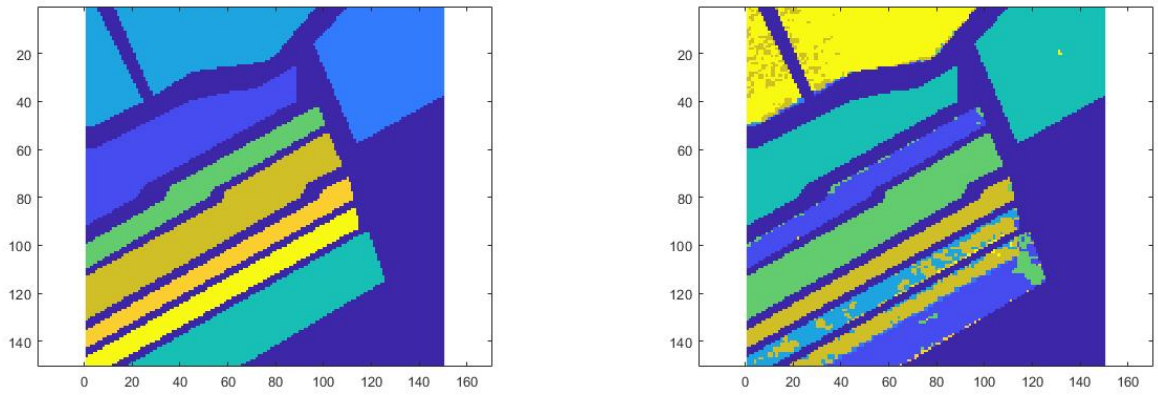

Figure 74: Complete-link with 8 clusters - valley image (left: the real clusters / right: the predicted clusters)

The confusion matrix proves that some clusters are well defined, such as cluster 6 and cluster 7, but others are completely wrong, such as cluster 1 and cluster 5. Also there are clusters that contain data points from two different categories, such as cluster 8. The total accuracy score is 65%, which means that the algorithm did not achieve better performance than other cost function optimization algorithms. The silhouette score for most of the clusters is high.

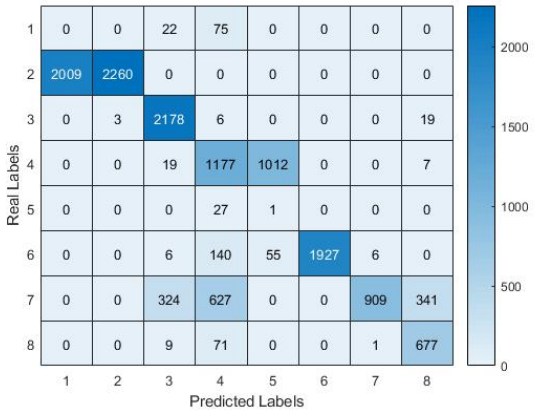

Figure 75: Confusion matrix of single-link

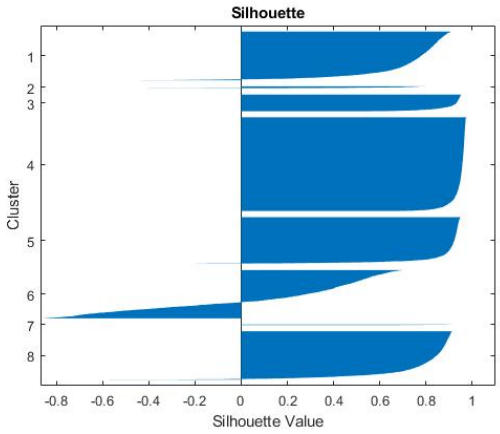

Figure 76: Silhouette score for single-link

5.2 Weighted Pair Group Method Centroid algorithm

The only difference between the *Weighted Pair Group Method Centroid algorithm* (WPGMC) and the previous one is the calculation of the distance $d(C_q, C_s)$. In this case, the distance is calculated using the formula:

$$d(C_q, C_s) = \frac{1}{2}d(C_i, C_s) + \frac{1}{2}d(C_j, C_s) - \frac{1}{4}d(C_i, C_j)$$

Each cluster is represented by a centroid and the algorithm calculates the distance of the weighted centroids. In particular, the $\vec{m_k}$ is the centroid of the cluster C_k:

$$\vec{m_k} = \frac{1}{n_k} \sum_{\vec{x} \in C_q} \vec{x}$$

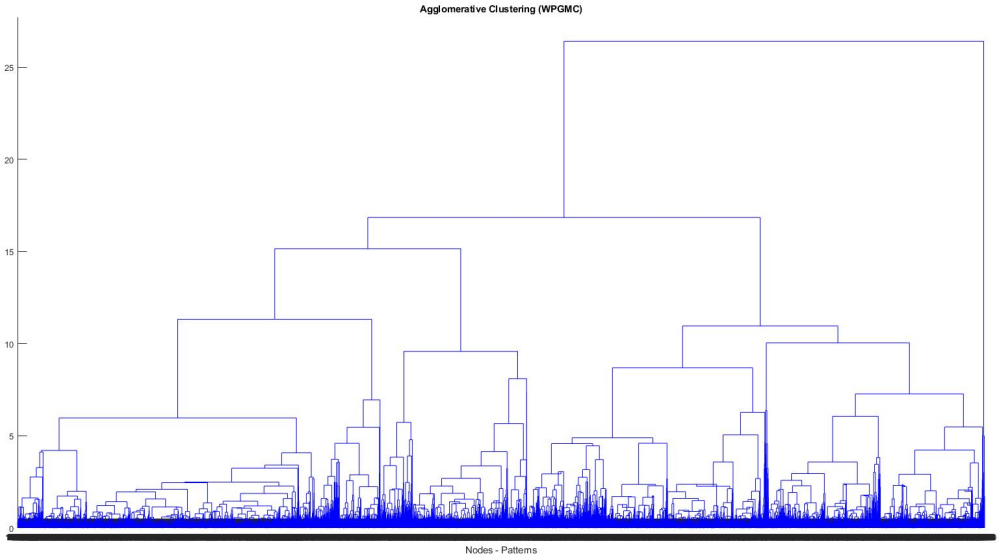

Figure 77: Dendrogram, using WPGMC algorithm

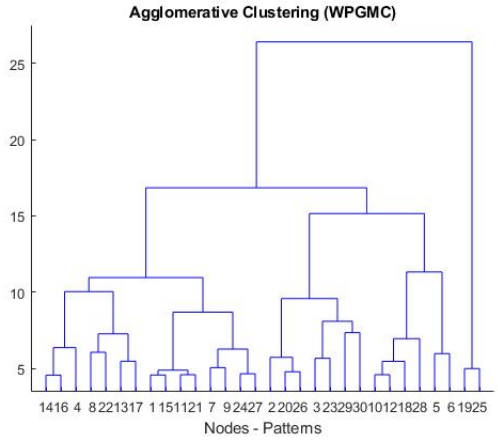

Figure 78: Dendrogram, using WPGMC algorithm (simplified)

We construct the dendrogram of the algorithm and we execute it with a different number of clusters as a parameter. Based on the dendrogram, we consider that a suitable number of clusters may be the 6 or the 8 and we execute the algorithm with both of them.

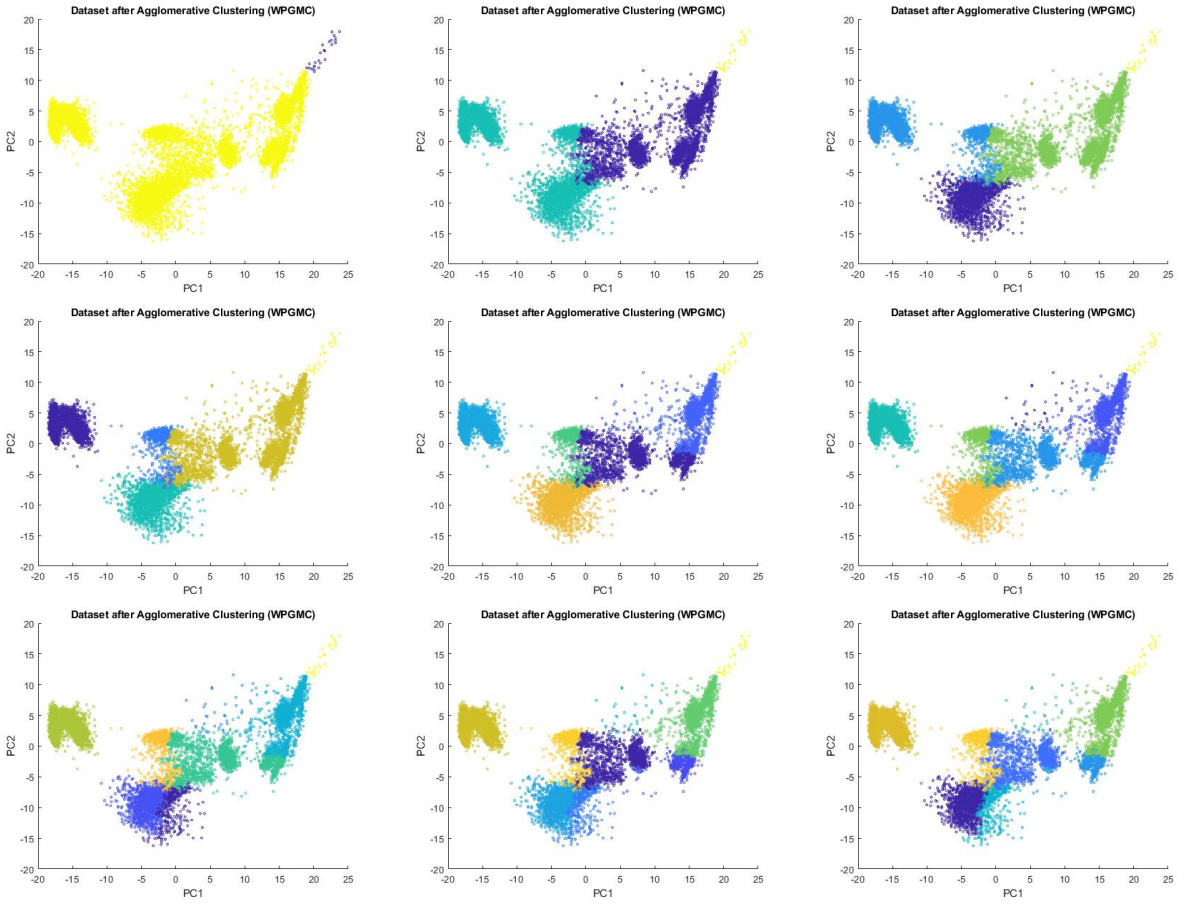

Figure 79: Clusterings that are produced using WPGMC linkage. Each clustering contains a different number of clusters

5.2.1 WPGMC with 6 clusters

We execute the clustering algorithm, setting 6 as number of clusters and WPGMC as linkage. The results are below.

The areas that refer to the clusters "Broccoli 1" and "Broccoli 2" are recognised as one crop. Also, the algorithm recognizes the cluster "grape", to a great degree. The clusters "lettuce 1" and "lettuce 3" are recognized, but they influence other clusters as well, such as the cluster "corn". As we can see in figure 80, almost all the predicted clusters do not correspond to the real clusters. Also, the silhouette score is not high for all of the clusters.

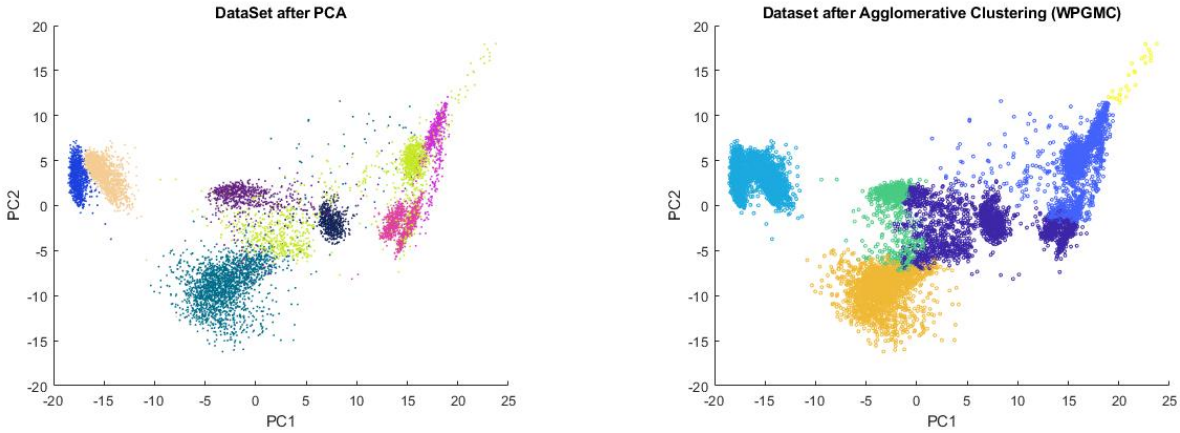

Figure 80: WPGMC with 6 clusters (left: the real clusters / right: the predicted clusters)

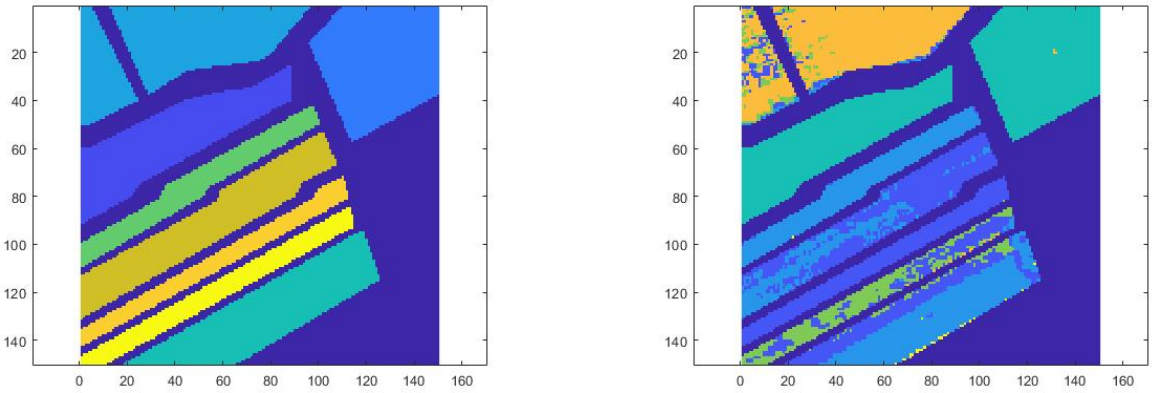

Figure 81: WPGMC with 6 clusters - valley image (left: the real clusters / right: the predicted clusters)

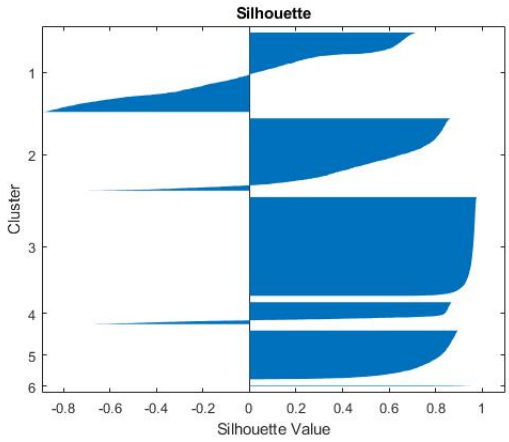

Figure 82: Silhouette score for WPGMC

5.2.2 WPGMC with 8 clusters

We execute the clustering algorithm, using the real number of clusters. The results are below.

The algorithm cannot recognize almost any of the real clusters. All the "predicted" clusters are a mixture of two or more "real" clusters, except from the "predicted" cluster that contains the clusters "Broccoli 1" and "Broccoli 2". If we study the confusion matrix, we will conclude that most of the predicted clusters do not correspond to the actual clusters, either because they do not contain any or almost any point of the expected cluster, such as cluster 7 either because they contain patterns that belong to a different cluster in reality, such as the clusters 2 and 4. The algorithm did not achieve to cluster the data points correctly and this is reflected in the total accuracy score, which is equal to 51%.

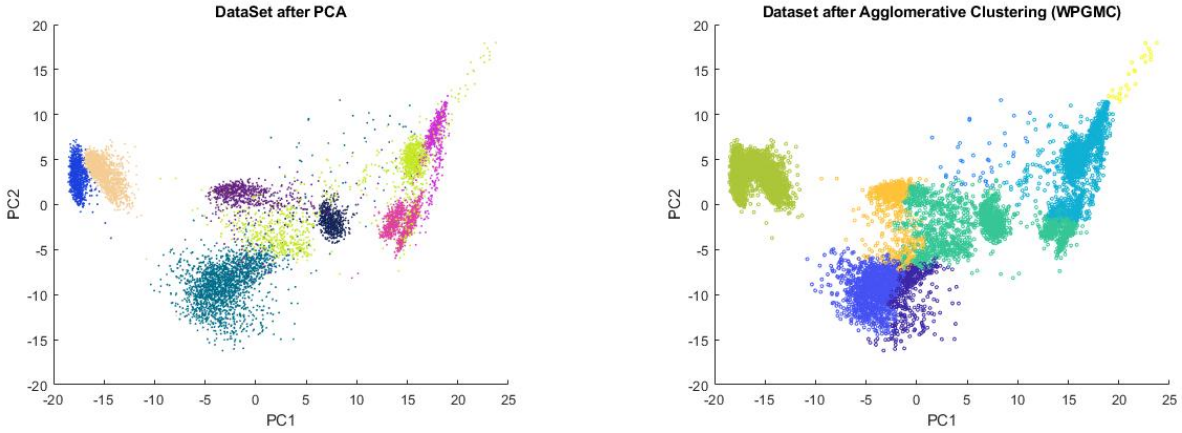

Figure 83: WPGMC with 8 clusters (left: the real clusters / right: the predicted clusters)

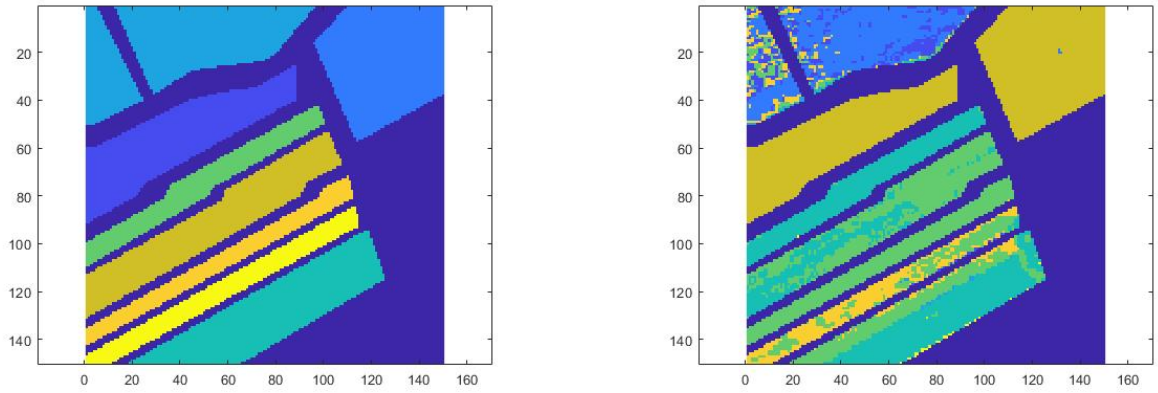

Figure 84: WPGMC with 8 clusters - valley image (left: the real clusters / right: the predicted clusters)

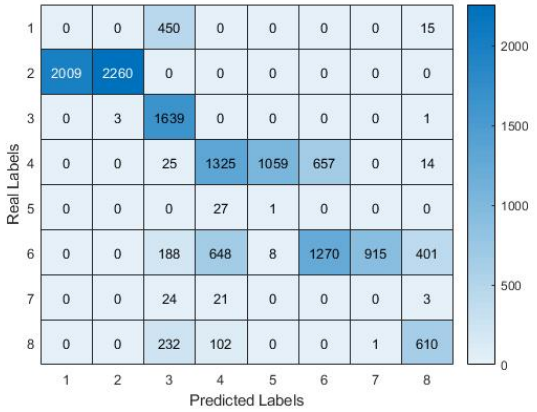

Figure 85: Confusion matrix of WPGMC

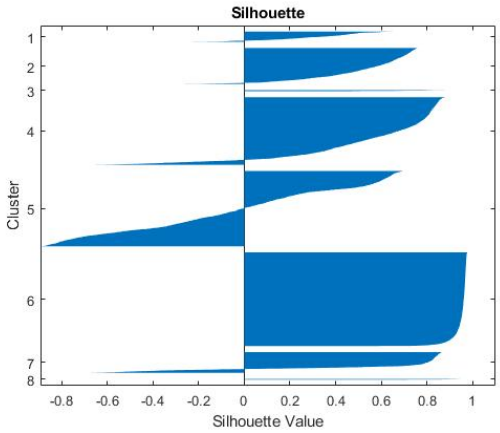

Figure 86: Silhouette score for WPGMC

5.3 Ward algorithm

The *Ward algorithm* uses a weighted Euclidean distance for the calculation of the distance between two clusters C_i, C_j:

$$d'_{ij} = \frac{n_i n_j}{n_i + n_j} d_{ij}$$

where $d_{ij} = ||\vec{m}_i - \vec{m}_j||^2$. Therefore, the distance between C_q and C_s is:

$$d'_{ij} = \frac{n_i + n_s}{n_i + n_j + n_s} d'_{is} + \frac{n_j + n_s}{n_i + n_j + n_s} d'_{js} - \frac{n_s}{n_i + n_j + n_s} d'_{ij}$$

This algorithm tends to merge clusters that have extremely different cardinalities, which means that one cluster has a much larger cardinality than the other one.

As always, we construct the dendrogram of the clustering, in figure 87, but we notice that most of the clusters merged in very low dissimilarities, compared to the dissimilarity of the final merges. This means that the algorithm recognized clusters from the very beginning of the execution. We adduce a simplified version of the dendrogram, as well, for presentation reasons. Based on the simplified dendrogram, we believe that a suitable parameter for the number of clusters may be number 6.

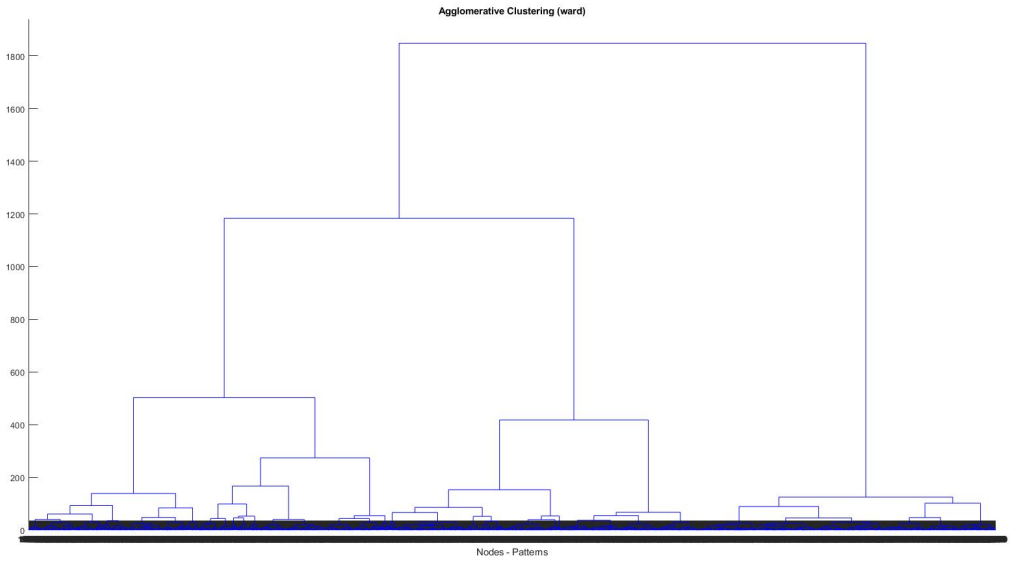

Figure 87: Full dendrogram, using ward algorithm

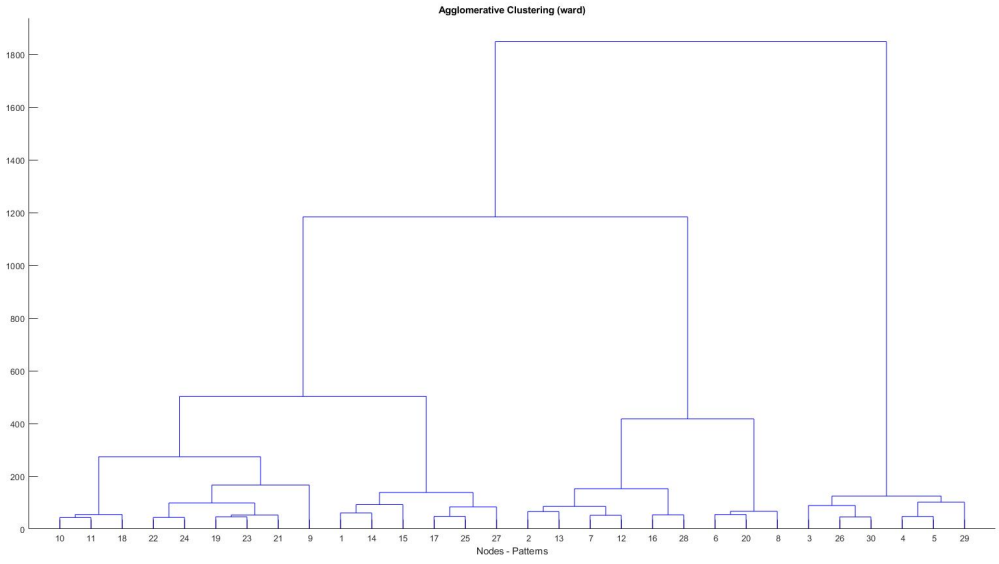

Figure 88: Simplified dendrogram, using ward algorithm

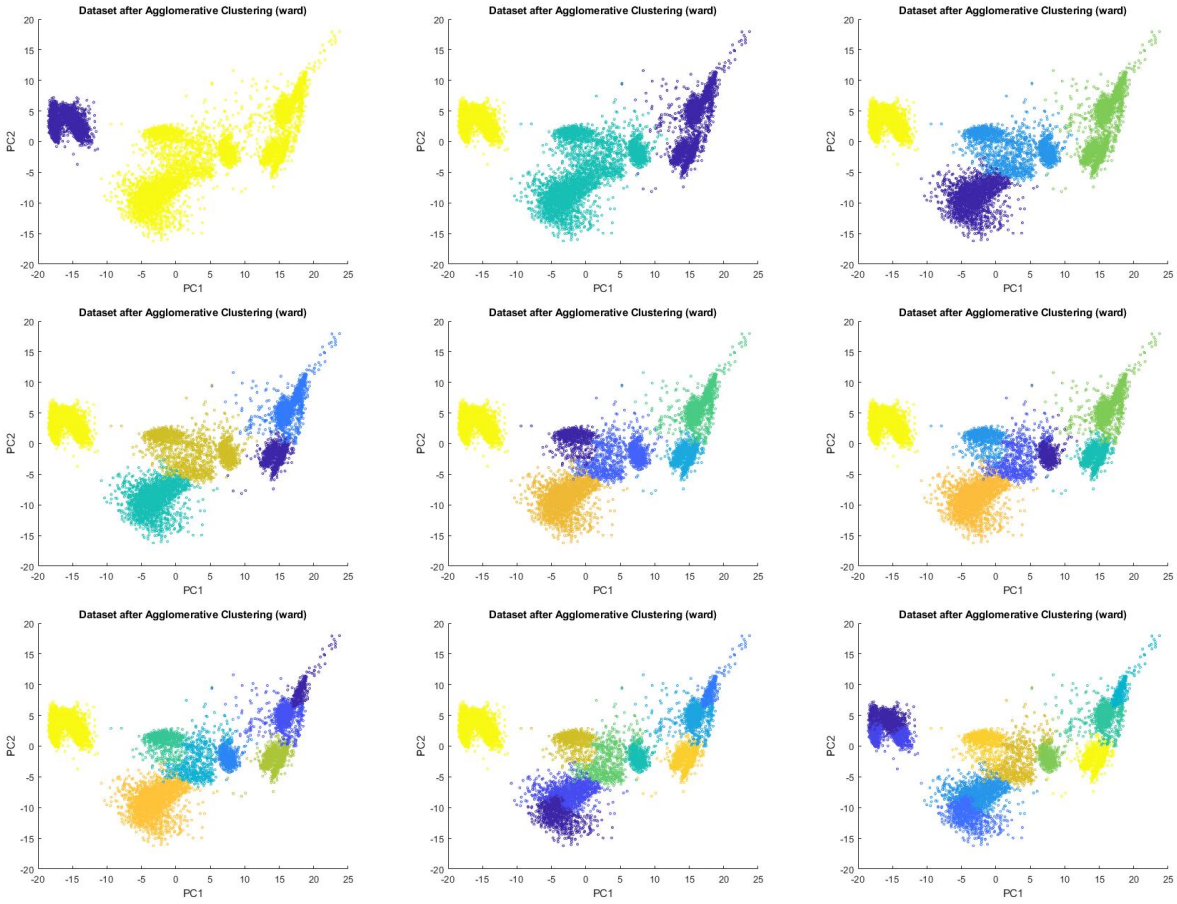

Figure 89: Clusterings that are produced using ward linkage. Each clustering contains a different number of clusters

5.3.1 Ward with 6 clusters

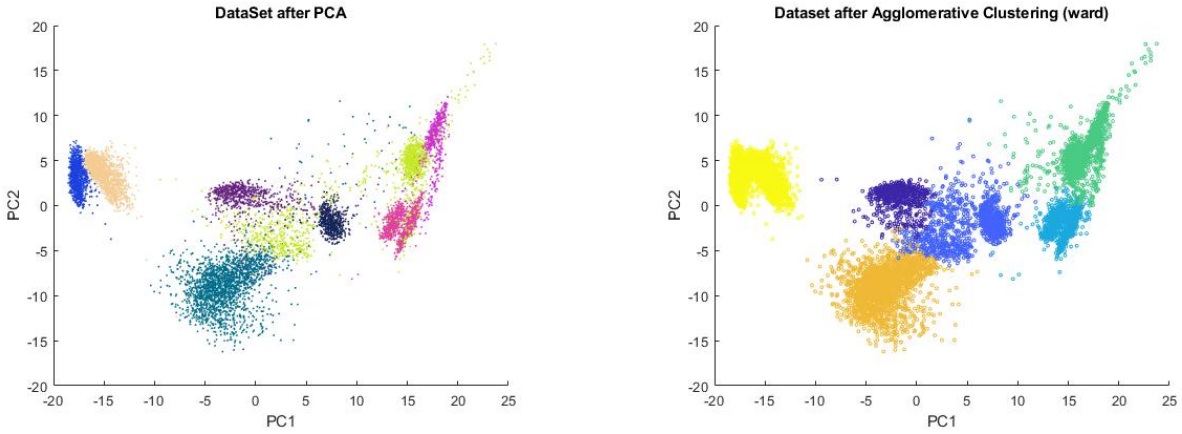

Figure 90: Ward with 6 clusters (left: the real clusters / right: the predicted clusters)

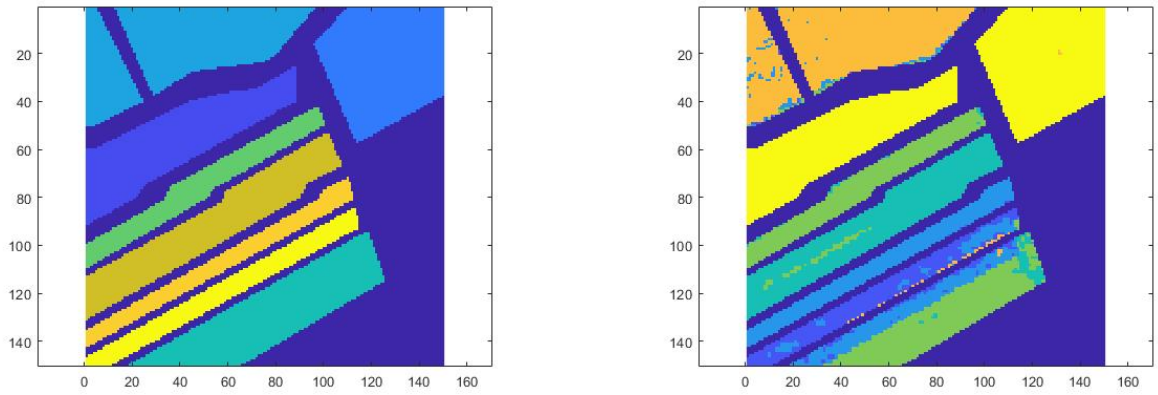

Figure 91: Ward with 6 clusters - valley image (left: the real clusters / right: the predicted clusters)

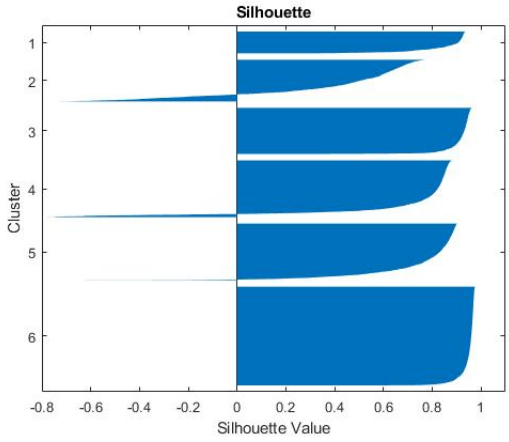

Figure 92: Silhouette score for Ward

We execute the agglomerative clustering using ward as linkage and 6 as number of clusters. We can notice that the algorithm recognizes most of the clusters, "grapes", "lettuce 1", "lettuce 2", "lettuce 3", it cannot divide the "broccoli" clusters into two different clusters and the corn cluster is a mixture of "lettuce 1" and "lettuce 3". If we check the results in figure 90, we will that there are three clusters (yellow, blue, green) that contain smaller actual clusters. So, we expect that if we execute the algorithm using a larger number as number of clusters, it will produce more accurate results.

5.3.2 Ward with 8 clusters

We execute the algorithm with 8 as number of clusters, because we know that this number is the real number of clusters, as well as, because in the previous execution we concluded that the algorithm may produce better clustering, using more clusters.

If we compare the figures 90 and 93, we will conclude that the algorithm "divides" the green and the blue cluster from the first figure and it creates four clusters from these two clusters. These four clusters correspond, more or less, to the actual clusters.

In this point we should mention again that the algorithm is an agglomerative algorithm, which means that it does not "divide" clusters, but it merges them, because the clustering of figure 93 is ahead of the clustering in figure 90. However, we believe that this presentation is most friendly for the reader.

Based on these results, we wonder if the execution with a larger number of clusters will produce better results. In particular, we wonder if the algorithm can recognize that the yellow cluster is two different clusters, in reality. However, if we check the figure 89, we that algorithm does not understand that the yellow cluster contains two different clusters. So, we do not experiment with a higher number of clusters.

The confusion matrix indicates that the majority of the points that belong to the actual cluster i are assigned to the predicted cluster i. The total accuracy is equal to 73%, which is the highest accuracy score for the hierarchical clustering algorithms. The silhouette score is high for most of the clusters and the valley figure is similar to the real one, except for the corn area. The broccoli areas are recognized as one cluster, as usual.

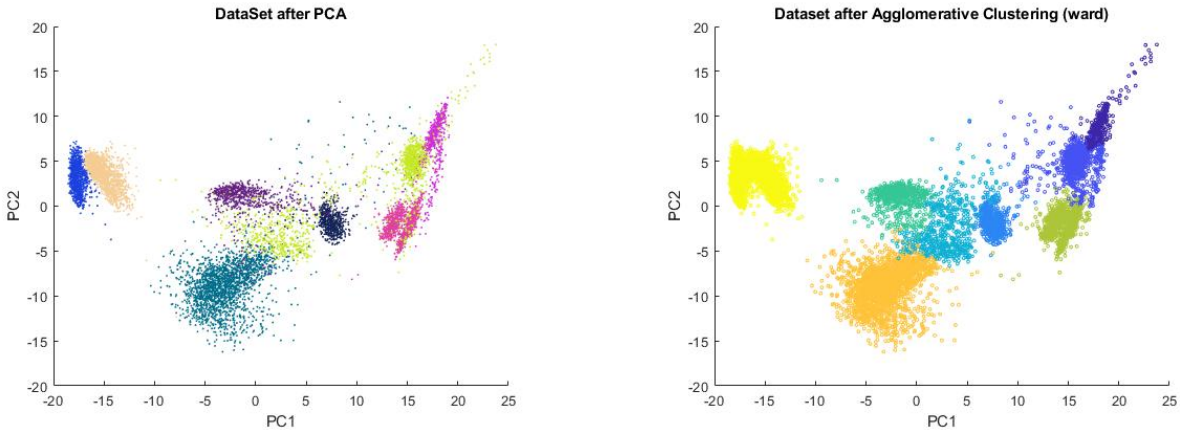

Figure 93: Ward with 8 clusters (left: the real clusters / right: the predicted clusters)

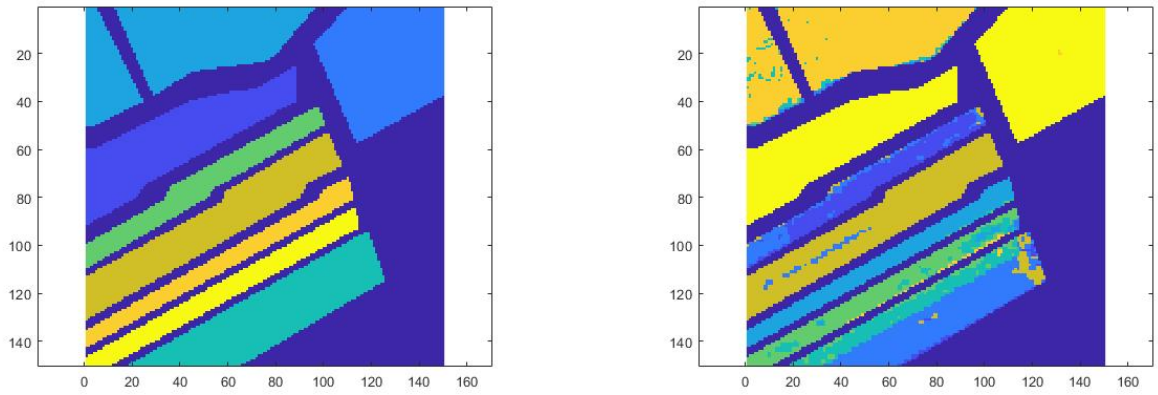

Figure 94: Ward with 8 clusters - valley image (left: the real clusters / right: the predicted clusters)

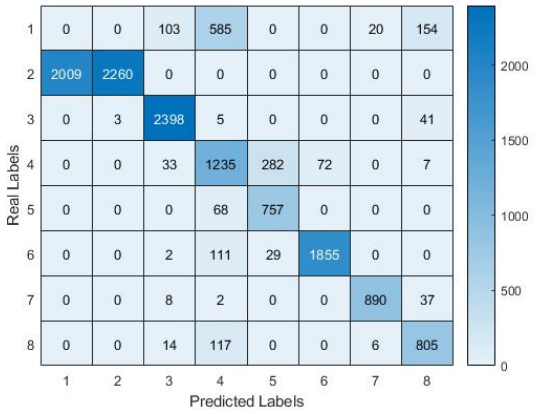

Figure 95: Confusion matrix of Ward

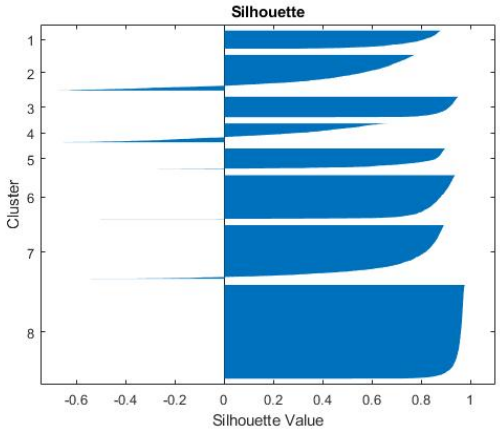

Figure 96: Silhouette score for Ward

6 Conclusions

We executed different clustering algorithms that follow completely different approaches (CFO /agglomerative). We examine the accuracy scores of each algorithm. The table below proves that the CFO approach was more suitable for this clustering problem. All of the CFO algorithms achieved decent accuracy scores. On the other hand, the agglomerative algorithms did not predict the actual clusters as we would expect, apart from the ward-linkage approach. This difference in the accuracy scores is reasonable, because the CFO algorithms do many iterations in order to create a clustering and in each iteration the clusters are modified. The agglomerative approach does not have this feature. It creates one clustering per iteration and if it produces a clustering that does not correspond to the real one, this "wrong" is transferred to the clusterings that follow. So, these algorithms do not correct their faults and this may lead to bad accuracy scores, as in this case.

Clustering Algorithm	**Accuracy**
k-means	69%
fuzzy c-means	74% ($q = 1.5$), 69% ($q = 2$)
possibilistic k-means	68%
probabilistic k-means	83%
agglomerative algorithm with complete linkage	65%
agglomerative algorithm with WPGMC linkage	51%
agglomerative algorithm with ward linkage	73%

The majority of the previous algorithms did not achieve to recognize the clusters in the red cycles in the figure below. The left two clusters correspond to the clusters "Broccoli 1" and "Broccoli 2" and the lime cluster (in the black cycle) corresponds to the "Corn" cluster. The "Broccoli" clusters are compact, but they are close in the upper area, so algorithms, such as the k-means cannot recognize them as two different clusters, because the mean confuses the algorithm. Many times the "Broccoli" clusters are considered to be one cluster by the algorithms. Additionally, the clusters in the right red cycle confuse many of the algorithms. Generally, the lime cluster, which corresponds to the "Corn" cluster is not compact and is distributed to the space. The clustering algorithms assigned many of the corn data points to different clusters. This is reflected in all the valley figures that we produced. The corn area always contains pixels from different crops. So, neither the agglomerative, nor the CFO approach achieved to cluster these patterns to the actual cluster.

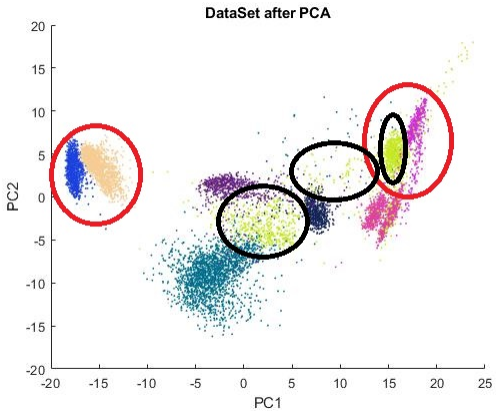

Figure 97: Clusters that were not recognised by the majority of the algorithms

In this point, we should emphasize that the majority of the techniques that we used in order to predict the most suitable number of clusters (we supposed that we do not know the actual labels of the patterns) proves that the data set may contain 6 physical clusters. This is reasonable, because the "Broccoli" clusters may be considered as one total cluster and the cluster "Corn" is distributed to a great degree, so one may conclude that the corresponding patterns belong to the densest algorithms.

If we examine only the accuracy scores of the CFO approach, we will conclude that the probabilistic k-means produces the best results and after this, the fuzzy c-means. The probabilistic k-means was the most time-consuming algorithm. It does many iterations in order to produce a clustering. In this point, we should mention that this algorithm achieved to recognize the "Broccoli 1" and "Broccoli 2" clusters that "tortured" the other algorithms. The fuzzy c-means did not achieve to recognize the "Broccoli" clusters and it divided them in the wrong way. The fuzzy approach with a low q value has better performance than the hard scheme, because it permits some of the patterns to belong to two or more clusters and this makes the algorithm more flexible. However, if we use larger q values, the algorithm cannot discover if the data set contains clusters, as we noticed in figure 20. The possibilistic k-means with a low q value discovered the clusters that contain the larger number of points. This is because the algorithm searches for the dense regions. So, the corn cluster was merged with the nearest clusters and the densest areas attracted more representatives. The results were not disappointing, but we consider that this algorithm is a stronger tool as a technique for recognizing the suitable number of clusters, than a clustering tool.

On the other hand, the range of the performance of the hierarchical clustering algorithms is large. The worst performance corresponds to the algorithm, which uses the WPGMC linkage. However, the ward linkage produced way better results. This is because the ward linkage tends to merge clusters that have extreme differences in the cardinalities and in our case, the clusters do not have the same size, so this feature helps to produce better results. Finally, the complete-link approach gives good accuracy score, but it does not discover all the actual clusters correctly. As we have already mentioned, this approach tends to find clusters with the same diameter and in this case we do not have clusters with the same size.

To sum up, the majority of the aforementioned algorithms created decent clusterings, but almost all the algorithms had a difficulty in predicting particular clusters. If we had to select only one algorithm from each approach, we would definitely have chosen the possibilistic k-means and the agglomerative algorithm with the ward linkage, not only because the corresponding accuracy scores were higher than other, but because these two algorithms achieved to discover the "Broccoli" and "Corn" clusters and in the same time the silhouette score for each cluster was high. If anyone studies the clustering results from these two approaches, without knowing the actual clusters, he/she will make safe conclusions about the data set. However, a complete study requires the execution of many different algorithms, so as to make safe conclusions and the "bad" clusterings help us to discover the "good" clusterings. Therefore, a complete study should contain the execution of many different algorithms.