Clustering Algorithms — Final Project

Konstantinos Konstantinidis

January 17, 2025

'Μη είναι βασιλικήν άτραπον επί γεωμετρίαν'
— Ευκλείδης

Contents

1 Introduction

Hyperspectral imaging is a powerful remote sensing technology that captures detailed spectral information across numerous wavelengths. Unlike traditional RGB imaging, which captures three bands (red, green, and blue), hyperspectral images can contain hundreds of contiguous spectral bands. This high-dimensional data provides much more detailed information about the materials and features present in a scene.

One of the key challenges in analyzing hyperspectral images is the inherent complexity of the data. The large number of spectral bands and the spatial relationships between pixels create high-dimensional data that can be difficult to interpret directly. To make sense of this complexity, machine learning techniques such as clustering are often employed. Clustering groups pixels or objects in an image based on their spectral similarity, without prior knowledge of the data's labels. This enables the identification of meaningful patterns, such as distinct material types, vegetation classes, or land cover features.

By grouping pixels with similar spectral characteristics, clustering helps in identifying homogeneous regions. However, clustering hyperspectral data also presents several challenges. The curse of dimensionality, due to the large number of spectral bands, can lead to sparse data representation and computational inefficiency. Additionally, the spectral similarity between different materials or features may not always be straightforward, requiring sophisticated algorithms to properly cluster the data.

1.1 The dataset

The dataset consists is a Hyper Spectral Image which depicts the Salinas Valley in California, USA. It has a 150 by 150 spatial resolution with 204 spectral bands. Thus the total number of pixels is 22500, stemming from eight ground- truth classes: one "corn", two types of "broccoli", four types of "lettuce", and on type of "grapes". The ground truth classes are displayed on the following figure (Figure 1), where each type of crop is colored in a different color, while the dark blue pixels have no ground truth information associated with them.

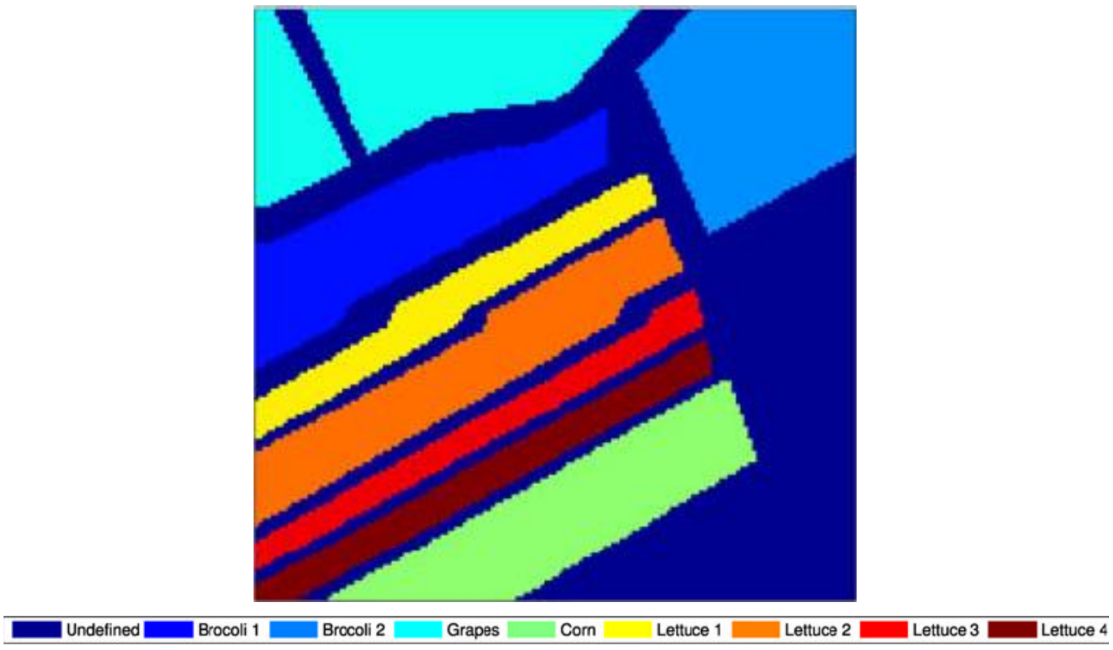


Figure 1: Ground truth, with labeled types.

1.2 Cost function optimization algorithms

This project will explore two families of clustering algorithms: cost function optimization clustering algorithms and hierarchical clustering algorithms. The algorithms we will test from the cost function optimization family include the k-means, fuzzy c-means, possibilistic c-means, and probabilistic (where each cluster is modelled by a normal distribution) algorithms. The cost function optimization algorithms cluster the data by minimizing (or maximizing) a loss function (or gain function).

1.2.1 k-means clustering algorithm

The k-means algorithm adopts the squared Euclidean distance to measure the dissimilarity between vectors $\boldsymbol{x}$ and cluster representatives $\boldsymbol{\theta}$. Each cluster is represented by a point representative. One major advantage of the k-means algorithm is its computational simplicity, making it suitable for processing large datasets. However, although this algorithm is fast and computationally efficient, it is sensitive to outliers.

1.2.2 fuzzy c-means clustering algorithm

The fuzzy c-means algorithm is very similar to the k-means algorithm with one major difference: the assignment of each point to clusters is not exclusive. Instead, each point is assigned to all clusters to varying degrees, represented by a membership function. Like the k-means algorithm, the clusters are represented by point representatives, it is fast and efficient for large datasets, but it remains sensitive to outliers.

1.2.3 possibilistic c-means clustering algorithm

The possibilistic c-means algorithm builds upon fuzzy c-means by introducing a degree of membership for each point in a cluster, which is determined independently of other clusters. This means that instead of enforcing that the sum of membership degrees across all clusters equals 1 (as in fuzzy c-means), each cluster's membership degree is calculated based on its own data points. This approach makes the possibilistic c-means algorithm more robust to noise and outliers, as points with low membership values can be excluded from cluster formation.

1.2.4 probabilistic clustering algorithm

Probabilistic clustering algorithms assume that data points are generated from a mixture of probability distributions, with each cluster corresponding to a different distribution. The algorithm estimates the parameters of these distributions (e.g., mean and covariance for gaussians) and assigns points to clusters based on their likelihood of belonging to each distribution. Probabilistic clustering is highly flexible and can model complex cluster shapes, but it requires careful initialization and is computationally more intensive than other clustering methods.

1.3 Hierarchical optimization algorithms

The hierarchical algorithms tested include the Complete-Link, the Unweighted Pair Group Method Average (UPGMA), and the Ward (or minimum variance) algorithms. Hierarchical clustering algorithms aim to cluster (or un-cluster) each data point iteratively. They produce a hierarchy of nested clusterings and they involve N steps at most (where N is the size of the data set). At each step the clustering is produced by the clustering of the previous step.

1.3.1 Complete-Link clustering algorithm

The Complete-Link clustering algorithm, also known as the maximum linkage algorithm, calculates the distance between clusters as the maximum distance between any two points in the clusters. This method tends to create compact and spherical clusters but may be sensitive to outliers, as a single distant point can significantly influence the clustering outcome.

1.3.2 Weighted Pair Group Method Average (WPGMA) clustering algorithm

The WPGMA algorithm is an agglomerative hierarchical clustering method that determines cluster linkage by calculating the weighted average of pairwise distances between points in different clusters. In this method, weights are proportional to the sizes of the clusters, meaning larger clusters have a greater influence on the calculated average distance. WPGMA is computationally straightforward and accounts for cluster size, but like UPGMA, it assumes a constant rate of evolution across all clusters, which may not be suitable for datasets with varying evolutionary rates.

1.3.3 Ward clustering algorithm

The Ward clustering algorithm, also known as the minimum variance method, aims to minimize the total within-cluster variance at each step of the hierarchical clustering process. It does so by merging the pair of clusters that results in the smallest increase in the total variance. This method often produces compact, homogeneous clusters and is particularly useful for datasets where minimizing variance is a priority. However, it can be computationally demanding for large datasets due to the iterative calculation of variances.

2 Feeling the data

The Salinas dataset consists of a Hyper Spectral Image of the Salinas Valley, which is represented by a hyper cube with two spatial dimensions of sizes 150 by 150 and a spectral dimension of size 204. As a first step, the data is transformed to a two dimensional matrix, where each row correpsonds to a pixel, and each column to a spectral band, giving a resulting shape of 22500 by 204. This will allow for an easier manipulation and a more clear approach to handling this type of dataset.

As for the first step in grasping the way this dataset behaves, I opted to calculate and visualize the Spearman's correlation coefficient for all features (spectral bands). The resulting heatmap (Figure 2) is not as expected[1]. Since we are dealing with a continuous spectrum, it would be more reasonable for elements closer to the main diagonal to have high correlation, decreasing for spectra that are further away (ie. points away from the main diagonal). Other than the few bands that segment the image in rectangular regions the rest of the features are highly correlated, which indicates that it will not be harmfull to decrease the feature space significantly.

This type of correlation matrix leads me to believe that those specific bands with lower correlation with the rest of the bands, are either physically different (and thus carry different information, so they must be examined), or they are technical artefacts or noise and they could be discarded. Both cases will be examined in this section.

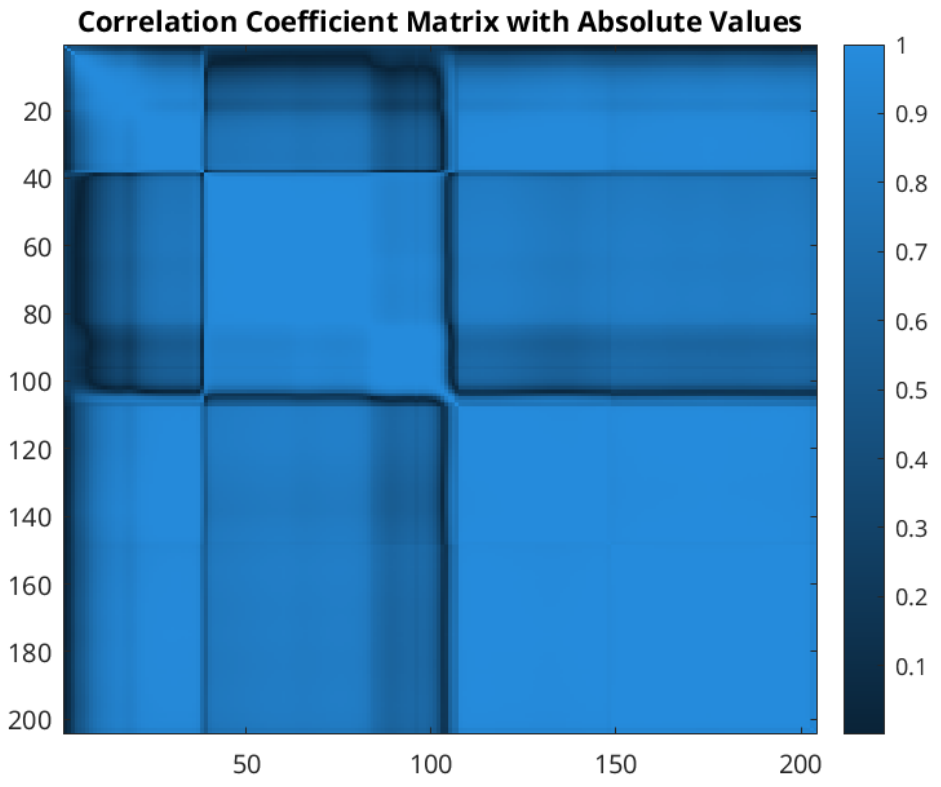


Figure 2: Heatmap of the absolute Spearman's correlation coefficients of the spectral data.

By plotting all combinations of the first six principal components of the feature space, against

[1]Note: it is the absolute value of the correlation coefficients that is plotted, as we only care about the absolute correlation between spectral bands at this point, and by plotting the absolute values, the figure is much more clearer and more easily interpretable

each other and coloring by their known types, we can get a feel for how they lie in their multidimensional space. Then by comparing those figures versus similar figures but with the data processed in some way, we can see if that specific processing was "successful" in the sense of not losing much information and succesfully reducing the feature space. The figure below (Figure 3) gives some information (especially when high ranking components are plotted), regarding the separation of the features in their high dimensional space, but that figure will prove even more valuable in the next steps, as it will allow us to compare how different preprocessing procedures will affect the data.

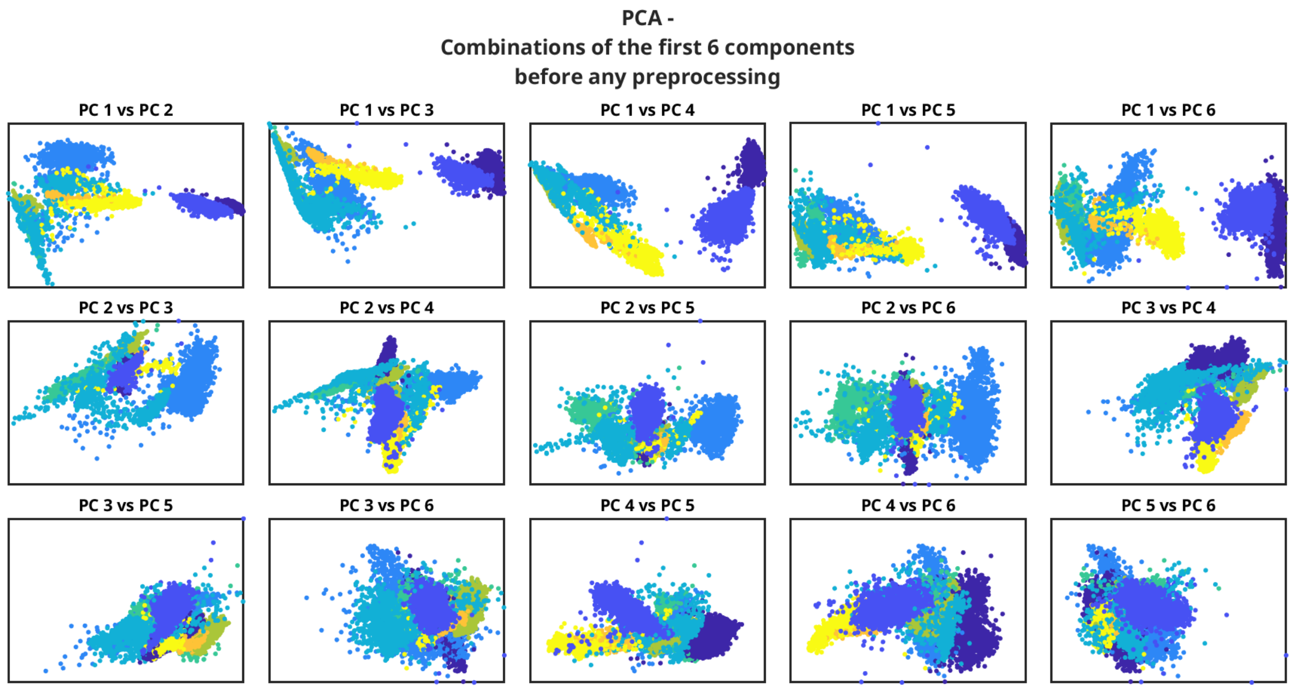


Figure 3: First six prinicpal components, plotted against each other and colored by type.

First, to check whether the specific aberrant spectral bands that are highly uncorrelated with the rest of the dataset are technical artefacts or not, I chose to keep the fourty spectral bands that had the highest total correlation coefficient and plot again the matrix PCA plot. Figure 4 does not seem (quantitatively at least) to diverge from the previous one. Some plots might seem different, but upon closer inspection it is obvious that they are the same, with some rotations or reflections which can happen after a PCA.

Next, I did the opposite, I kept the fourty spectra that had the lowest total correlation coefficient and created the same plot with the six highest principal components (Figure 5). The difference in how well the hyper dimensional points are separated is not substantial, but it is still significant. The principal components whose data consist of the most correlated features, seem to be better separated. This leads me to conclude that those bands that had very low correlation are technical artefacts and are not informative.

Since the feature vectors are describing spectral information, which can be considered as continuous, the method I chose to reduce the feature dimensions was to bin together bands and replace them by their mean. So for example, for the 204 bands, dividing them into six bins, the dimensions of the feature space would decrease 6-fold, and the new feature space would have 34 dimensions.

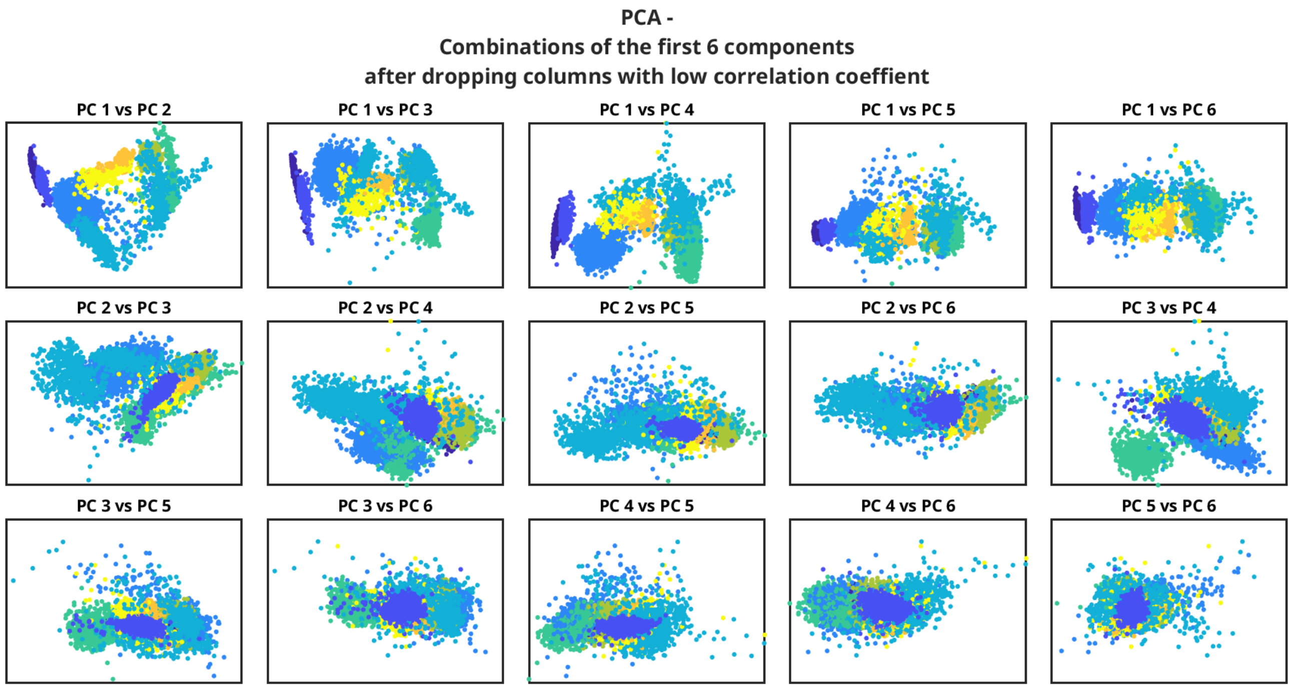


Figure 4: First six prinicpal components, plotted against each other and colored by type, after the features (bands) with high Spearman's correlation coefficient are selcted.

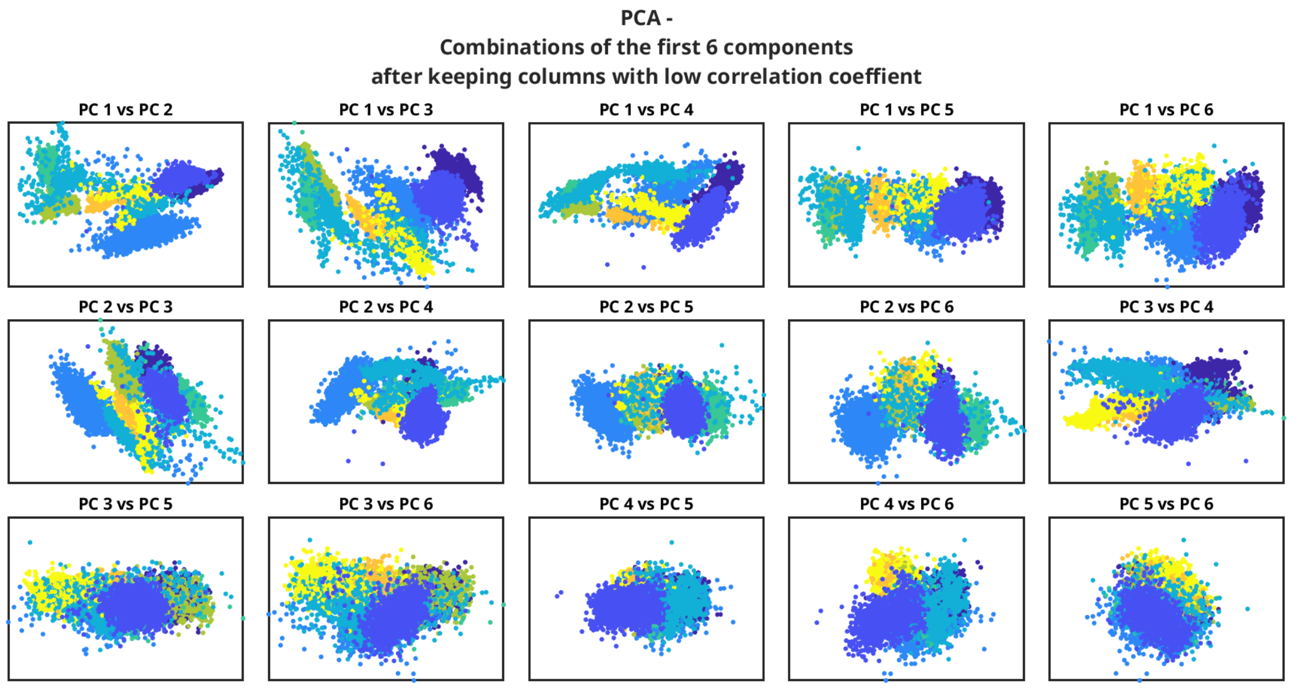


Figure 5: First six prinicpal components, plotted against each other and colored by type, after the features (bands) with low Spearman's correlation coefficient are selcted.

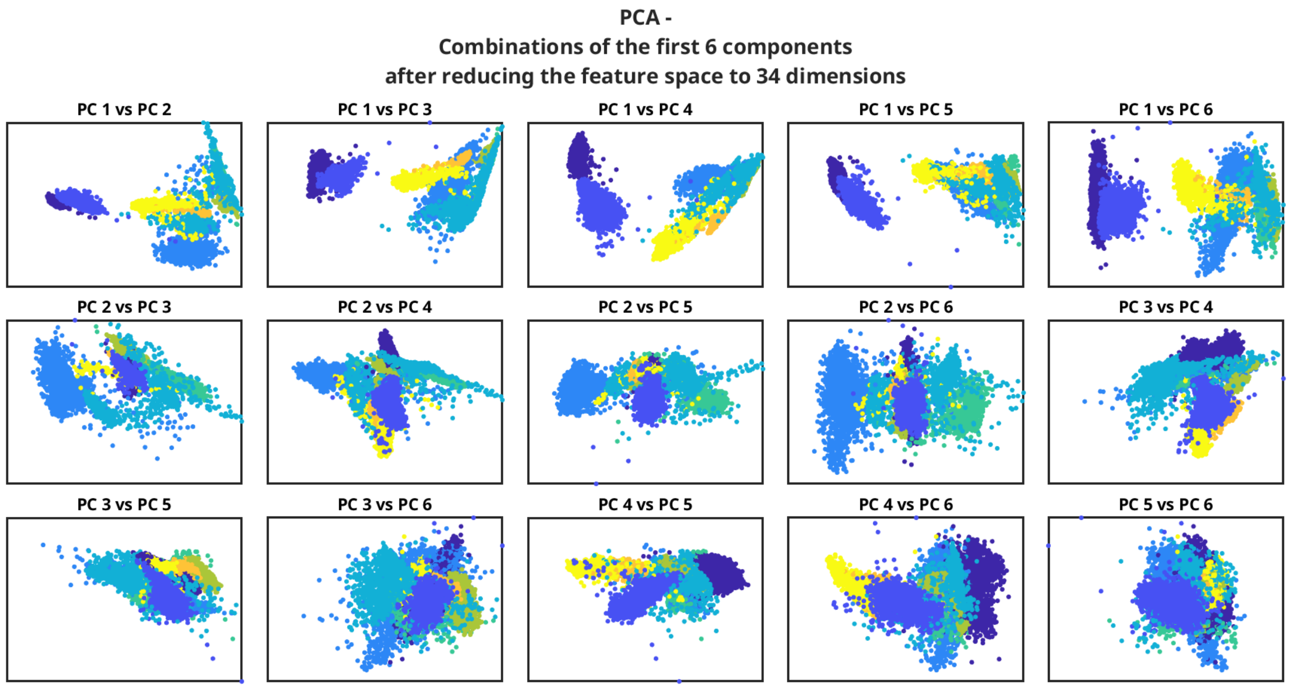


Figure 6: First six prinicpal components, plotted against each other and colored by type, after the features dimension were reduced to 34, with the binning method

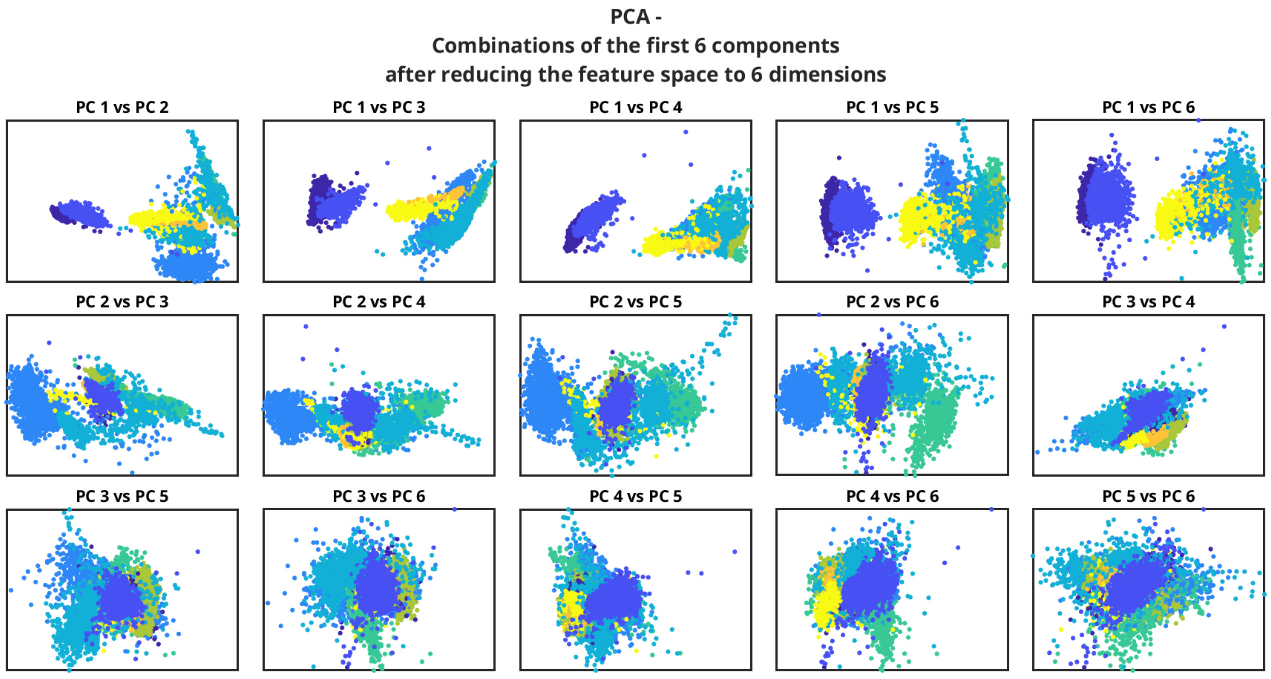


Figure 7: First six prinicpal components, plotted against each other and colored by type, after the features dimension were reduced to 6, with the binning method

This method of spectral binning with mean aggregation was implemented once for a 6-fold and once for a 34-fold reduction in feature dimensions and the resulting dataset was again plotted with the PCA method (Figure 6 and Figure 7). From the plots that were generated, we can see that even for a 34-fold reduction in the feature dimensions, the the higher ranking principal components are identical to the unprocessed data. In the case of the 34-fold reduction, as the rank of the principal components that are plotted increases, so does the separability of the points decrease, indicating that a smaller reduction in the feature dimensions might be optimal.

Before settling on the final method of preprocessing my dataset, I also plotted the correlation coefficient heatmap of the features with reduced dimensions by the binning method to check for the presence of uncorrelated features (Figure 8). Since there are some bands that have low correlation and I assume they are technical artefacts, the final preprocessing that I have chosen will include the removal of those low correlated bands, and then a spectral binning with mean aggregation.

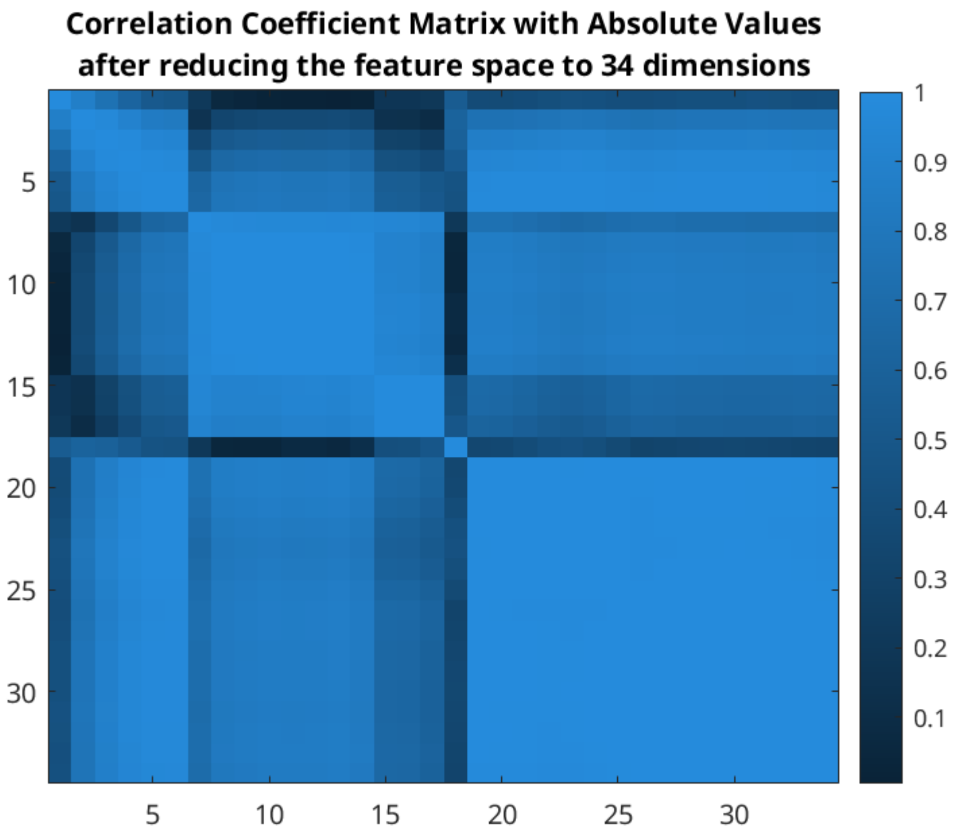


Figure 8: Heatmap of the absolute Spearman's correlation coefficients of the spectral data, after the spectral binning dimensionality reduction method.

3 Testing the clustering algorithms

3.1 Preprocessing

Before proceeding with the tests of all the different clustering algorithms, the data was preprocessed based on the findings mentioned above. First, the 42 least correlated features were discarded, leaving 160 bands. Then, using the binning method with mean aggregates the features were reduced by another factor of 16, leaving behind only 10 spectral bands. The resulting dataset is what was used for the analyses that follow.

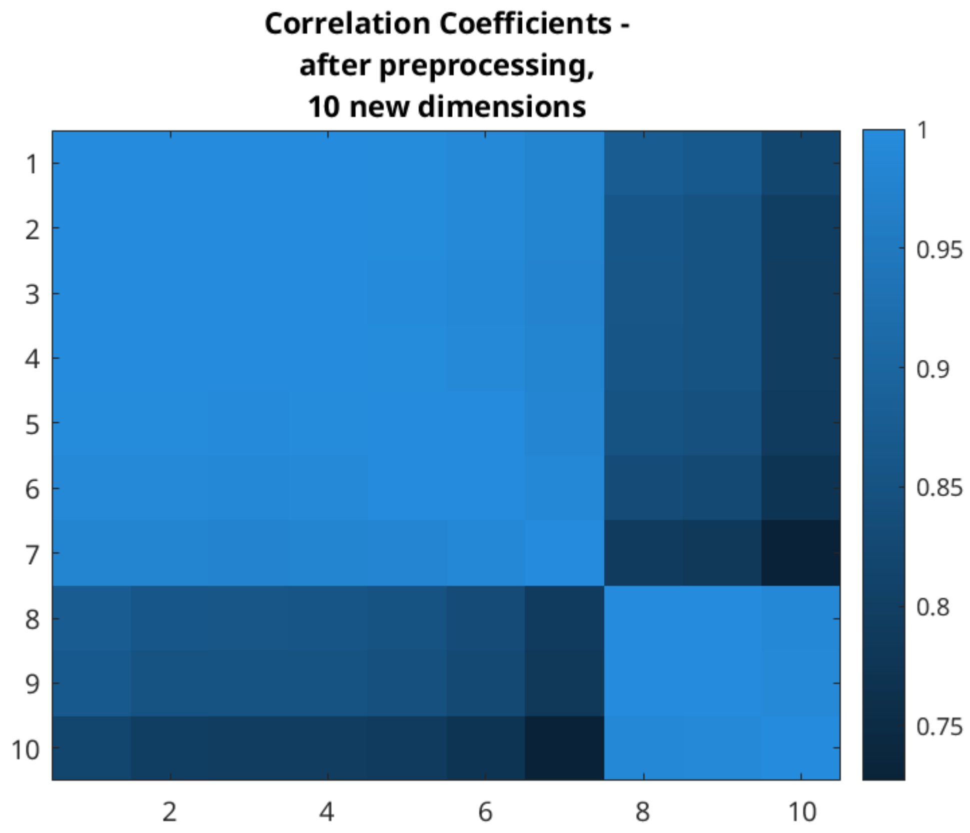


Figure 9: Heatmap of the absolute Spearman's correlation coefficients of the spectral data, after preprocessing.

Based on the heatmap of the Spearman's correlation coefficients (Figure 9), we can see that all correlations are above 0.75, and if the previous assumption —that the bands that are uncorrelated with the others are noisy due to technical artefacts— then the preprocessing was successful.

The next figure, Figure 10, shows the first six principal components of the preprocessed dataset plotted against each other. When compared with Figure 3 which depicts the same plot, but with data that are not preprocessed, it is visible that the high dimensional features closely resemble the structure of the unpreprocessed data. This similarity suggests that the preprocessing steps, while potentially improving certain aspects of the data, do not significantly alter the intrinsic relationships captured by the principal components. Thus, the underlying patterns remain consistent between the preprocessed and unpreprocessed datasets.

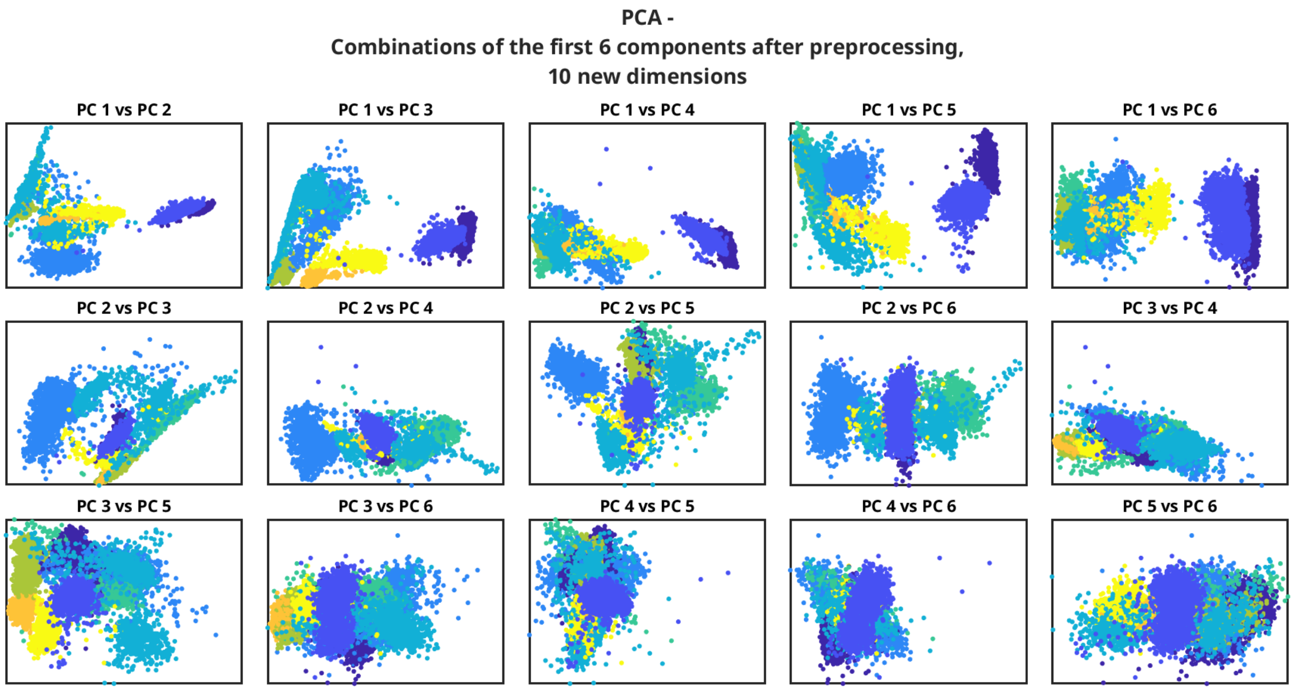


Figure 10: First six prinicpal components, plotted against each other and colored by type, after preprocessing.

3.2 k-means clustering algorithm

The first step was to test the algorithm for a range of different numbers of clusters m and plot the cost function J over that range of clusters to find the knee of that function.

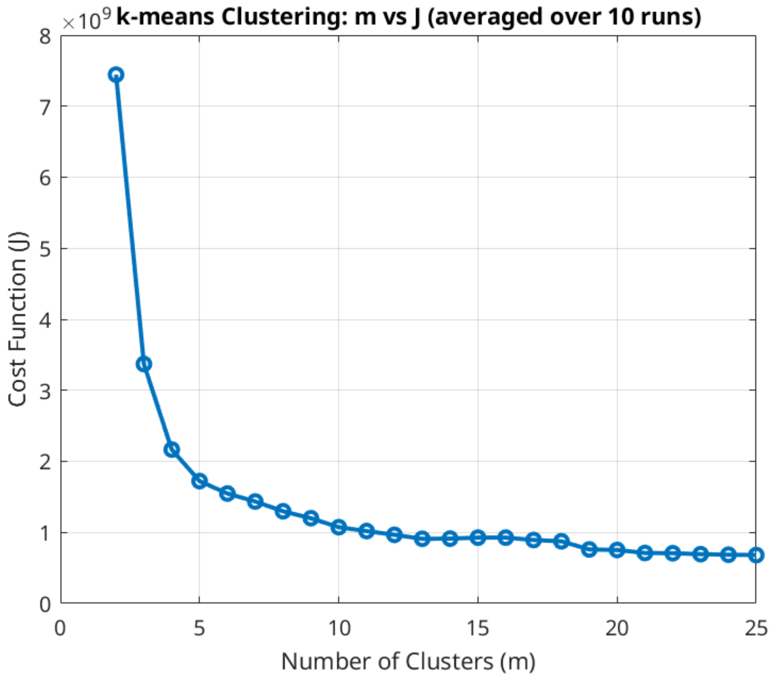
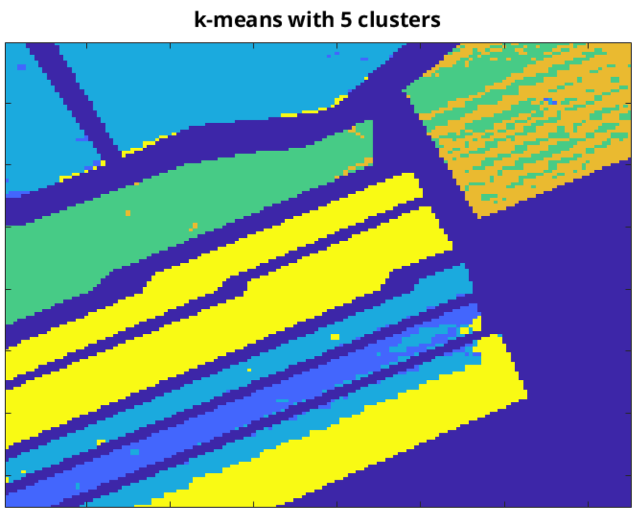
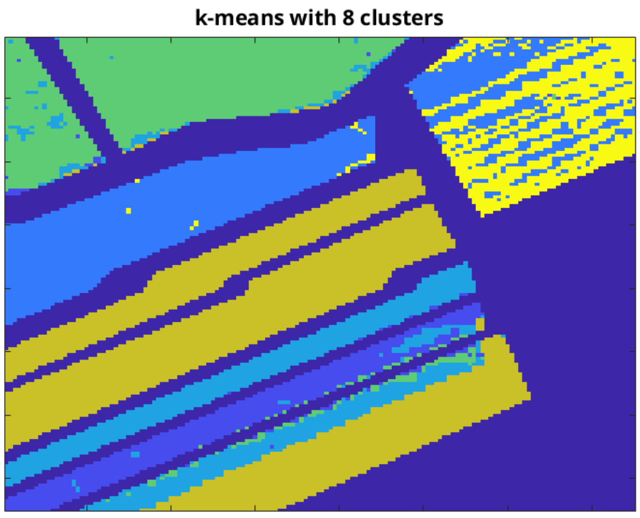


Figure 11: Cost function J over the number of clusters m. The apparent knee of the plot is 5.

From this point, I chose 5 as the number of clusters (even though the ground truth image shows that the number of physical clusters is 8). This was done to emulate a real clustering problem where the labels are not known *a priori*. The algorithm managed to cluster the data in less than a tenth of a second (for the unprocessed data it took around 3.5 seconds).



(a) 5 clusters.



(b) 8 clusters.

Figure 12: Comparison the k-means clustering algorithm with 5 and 8 clusters.

The resulting cluster labels are depicted above (Figure 12), as a way to compare them qualitatively with the ground truth image. Also the Adjusted Rand Index (ARI) and the Normalized Mutual Information (NMI) were calculated to give a quantitative metric to the problem. The output of the matlab script which includes these metrics is provided below.

```
k-means with 5 clusters
Adjusted Rand Index (ARI): 0.8166
Rand Index (RI): 0.9368
Mirkin Index (MI): 0.0632
Hubert Index (HI): 0.8736
Normalized Mutual Information (NMI): 0.8040
Elapsed time is 0.084712 seconds.

k-means with 8 clusters
Adjusted Rand Index (ARI): 0.8539
Rand Index (RI): 0.9511
Mirkin Index (MI): 0.0489
Hubert Index (HI): 0.9023
Normalized Mutual Information (NMI): 0.8265
Elapsed time is 0.089917 seconds.
```

The preprocessed data was also clustered using 8 clusters which is the actual number of physical clusters and also on the unpreprocessed data, for comparison. The results were almost identical to the ones above, meaning that the preprocessing was successful, and by using a the correct of clusters the gain in accuracy was minimal. **Note:** All additional plots that are not depicted here can be found in the zipped folder containing this pdf.

3.3 fuzzy c-means clustering algorithm

Like before, the algorithm was used with a range of clusters m and the resulting cost function J was plotted to find the knee (Figure 13).

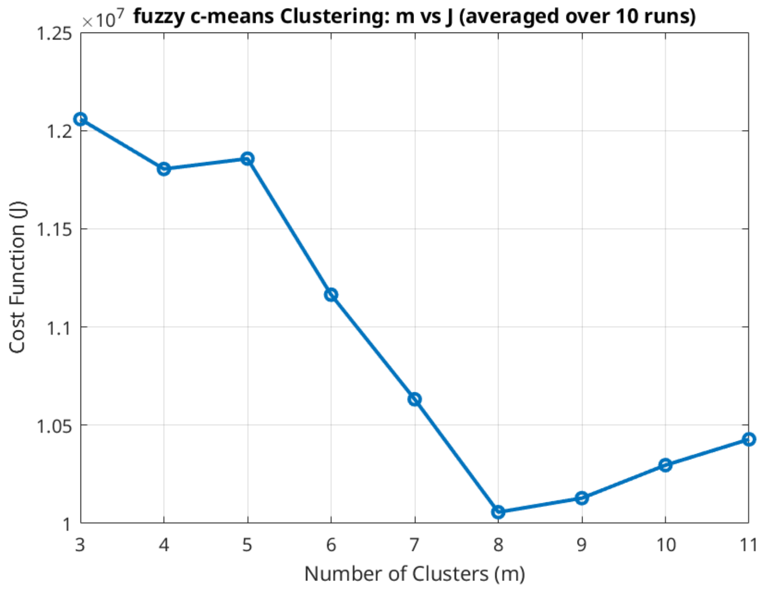


Figure 13: Cost function J over the number of clusters m. The apparent knee of the plot is 8.

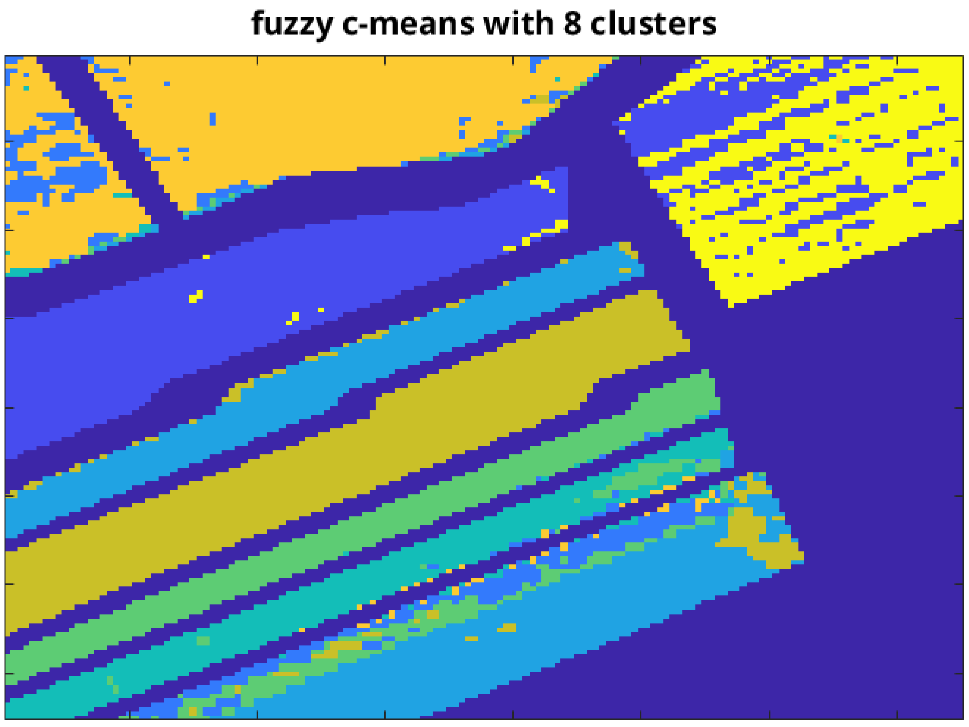


Figure 14: Fuzzy c-means clustering of the HSI, with 8 clusters

The resulting cluster labels are depicted above (Figure 14), as a way to compare them qualitatively with the ground truth image. The Adjusted Rand Index (ARI) and the Normalized

Mutual Information (NMI) were calculated to be at 0.9080 and 0.8504 respectively. The algorithm managed to cluster the data in 22.5 seconds.

3.4 possibilistic c-means clustering algorithm

The η variable for the possibilistic c-means algorithm was calculated with the following equation 1.

$$\eta = \frac{\frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{x} - (\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x_i})||^2}{q\sqrt{m}} \tag{1}$$

In addition to different numbers of clusters (m), I also decided to explore how q affects the cost function J and the resulting clusters. The possibilistc c-means cost function adjusted with η is shown below (equation 2). The following figure (Figure 15) shows the relationship of J with different values of m. The fact that J increases continually indicates that the algorithm did not manage to find the optimal clustering configuration, since if it had found it then we would expect to see a dip in the plot.

$$J_q(U, \Theta) = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij}^q d(\boldsymbol{x_i}, \boldsymbol{\theta_j}) + \sum_{j=1}^{m} \eta_j \sum_{i=1}^{N} (1 - u_{ij})^q \tag{2}$$

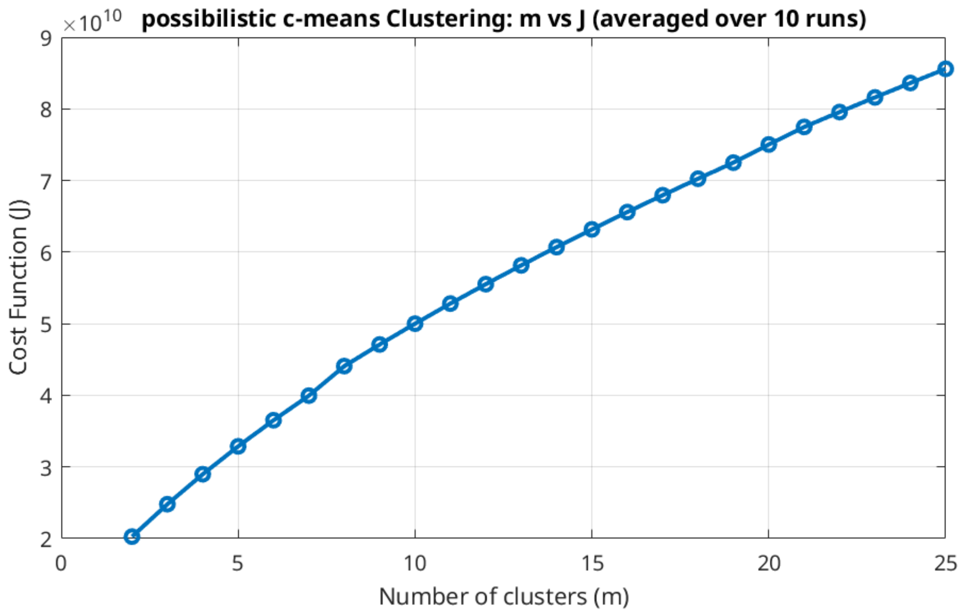
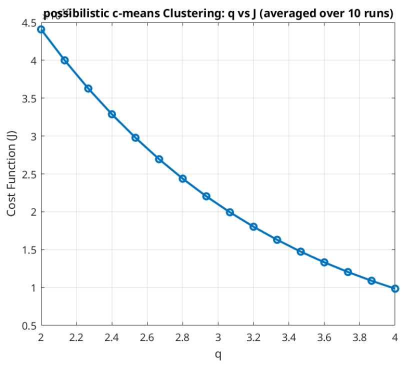


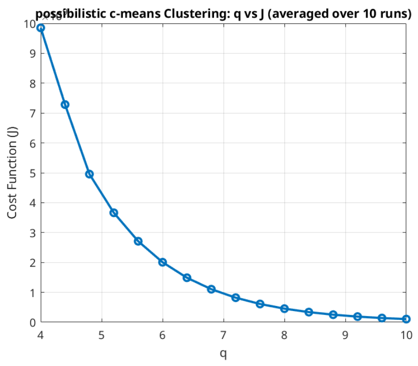
Figure 15: Possibilistic clustering algorithm, cost function J over clusters m.

As q increases, the influence of the membership values ${u_{ij}}^q$ in the first term of the cost function becomes less sharply defined. The second term involving $(1 - u_{ij})^q$ is related to penalizing low memberships. As q increases, this penalty becomes less strict, preventing outliers and noise from contributing to the optimization. This can enhance the robustness of the algorithm to noise but may also reduce its ability to tightly cluster well-defined data. In essence increasing q causes almost equal contributions of all vectors to all clusters which should lead to a decrease in the cost function J.

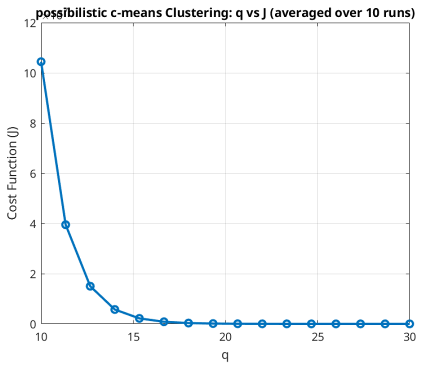
Below there are 3 plots (Figure 16) showcasing how the cost J decreases as q increases. Figure 16a has values ranging from 2 to 4, Figure 16b has q values ranging from 4 to 10, and Figure 16c has q values ranging from 10 to 30. Based on those plots, I decided to test algorithm for q equal to 2 and 12 to compare the results and showcase that although J decreases with larger a q value, the actual clustering should be worse.



(a) q from 2 to 4.

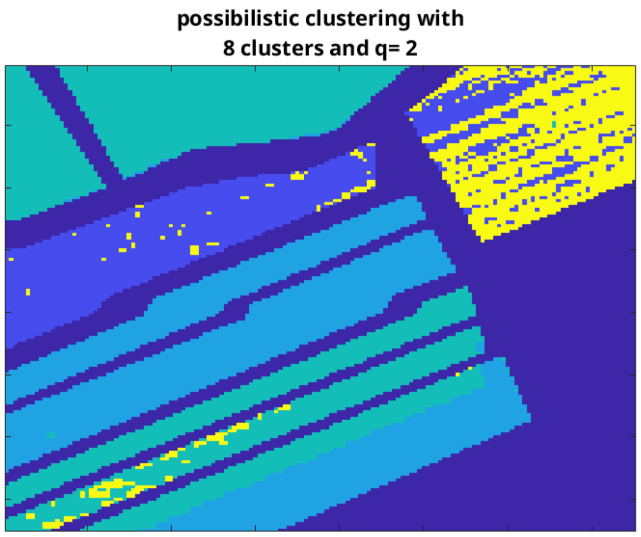


(b) q from 4 to 10.

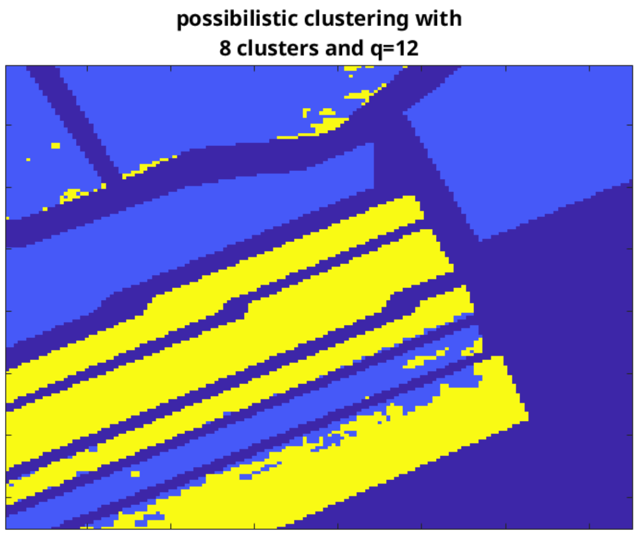


(c) q from 10 to 30.

Figure 16: Comparison of how small, medium and large values of q affect the cost function of the possibilistic c-means algorithm.



(a) $q = 2$.



(b) $q = 12$.

Figure 17: Comparison of how well q equal to 2 and 12 perform for the possibilistic c-means algorithm.

The algorithm completed after around 5 seconds for both cases of q's, while for q equal to 12 it was slightly faster. The resulting clusters are depicted above (Figure 17). The number of clusters m was constant and equal to 8 for all cases.

From the figure above, it is obvious that the lower q value helped the algorithm to perform better, although it still seems to be the worse performing one up to this point. The ARI and the NMI for these two clusterings is also the lowest so far, with an $ARI = 0.7869$ and $NMI = 0.7813$ for the case where q was 2 and an $ARI = 0.6287$ and $NMI = 0.6994$ for the case where q was equal to 12.

3.5 probabilistic clustering algorithm

When trying to execute this clustering algorithm without normalizing the data, I encountered several issues with the computations. This was because normalization of the dataset is a critical preprocessing step that significantly impacts the stability and performance of the algorithm. Without normalization, features with larger scales can disproportionately dominate the computation of distances and probabilities, leading to numerical instability and biased parameter estimates. Specifically, in the Expectation-Maximization (EM) step, the covariance matrices are sensitive to differences in feature scales. This transformation ensures that each feature contributes equally to the model's likelihood computation, improving both numerical stability and convergence behavior. Consequently, normalization is an essential preprocessing step in ensuring robust and reliable performance.

The probabilistic clustering algorithm was tested for a range of different clusters (m) to see if the cost function J would be affected (Figure 18). Since the function stays almost constant with only small fluctuations, it's indicated that the convergence threshold was set too high (in this test it was set to 0.5 for time efficiency). For the next clustering it will be lowered significantly and the number of clusters m will be set to 8.

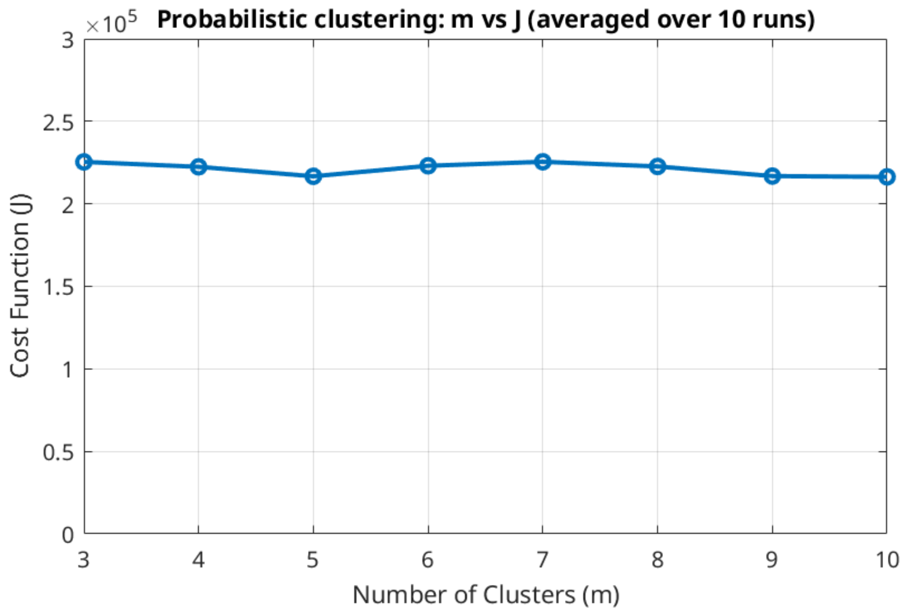


Figure 18: Probabilistic clustering algorithm, cost function J over clusters m.

The algorithm took around 12 seconds to complete, the Adjusted Rand Index was calculated to be 0.7719 and the NMI at 0.7374. When taking into account the speed of the algorithm and it's clustering performance, so far it is one of the worst performing. The clustering produced by this algorithm is provided below (Figure 19)

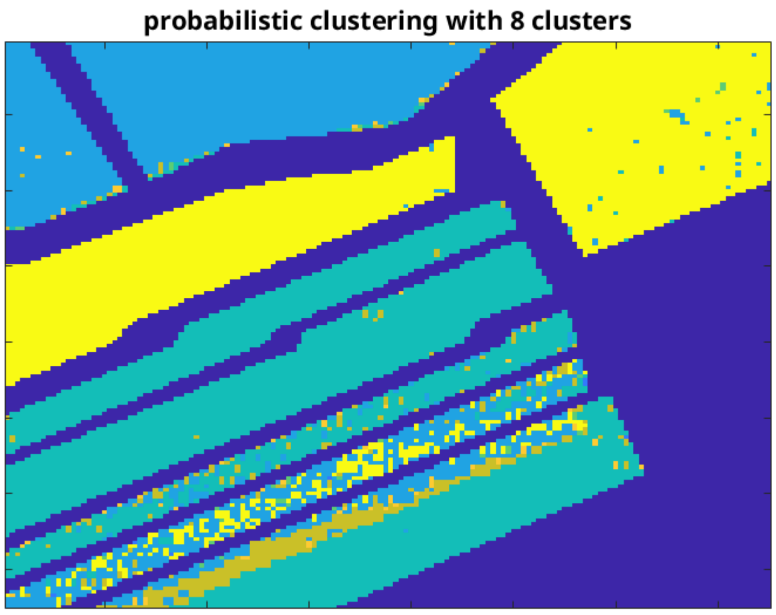


Figure 19: Probabilistic clustering algorithm with 8 clusters.

3.6 Complete-Link clustering algorithm

When using a hierarchical clustering method, one way to finding the optimal number of clusters is to compare all cluster lifespans. A cluster's lifespan is the proximity level at which it was "killed" (ie. a new data point was integrated in it or it was joined with another cluster) minus the proximity level at which it was "born" (ie. the level at which it was created). In the figure below (Figure 20) the 30 highest lifespans are displayed.

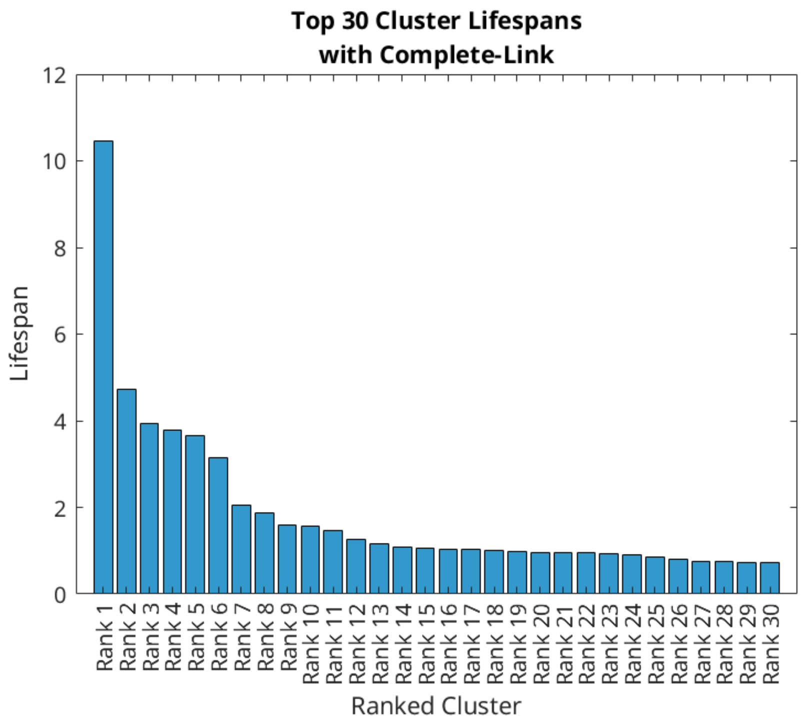
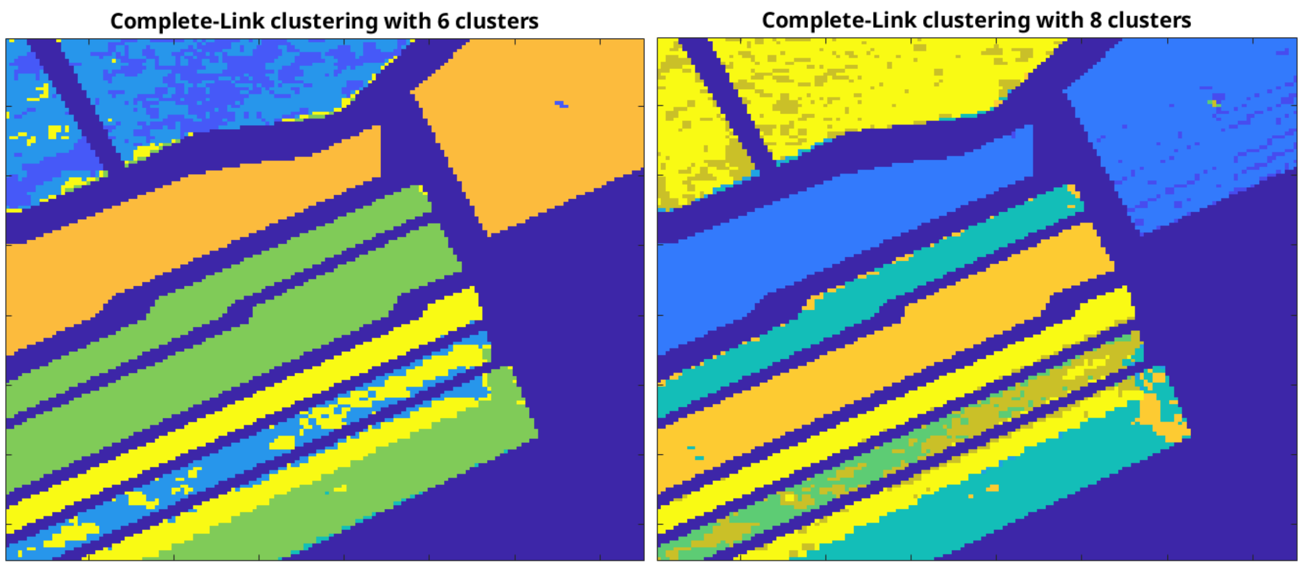


Figure 20: Lifespan of the clusters generated by the Complete-Link clustering algorithm.

From Figure 20 I decided that I will use 6 clusters (since using only 1 cluster does not make sense in a clustering problem). The Complete- Link clustering algorithm was then executed and it was cutoff at either 6 or 8 clusters. The ARI for the first case was 0.8101 and 0.8466 for the second, while the execution time was between 1 and 2 seconds for both cases. In the figure below (Figure 21) the Complete-Link clustering algorithm with a cutoff at the 6 final clusters is compared with the one at 8 clusters. Both clusterings seem good, while the one with 8 seems to differentiate the two plant types in the middle of the image better.



(a) 6 clusters.

(b) 8 clusters.

Figure 21: Comparison of the Complete-Link clustering algorithm with a cutoff at 6 and 8 final clusters.

3.7 Weighted Pair Group Method with Average (WPGMA) clustering algorithm

The WPGMA algorithm is tested in similar spirit. The figure below (Figure 22) shows the lifespans of the 30 clusters with with highest lifespans and from there we can see that the biggest “jumps” happen at 2 and 4 clusters.

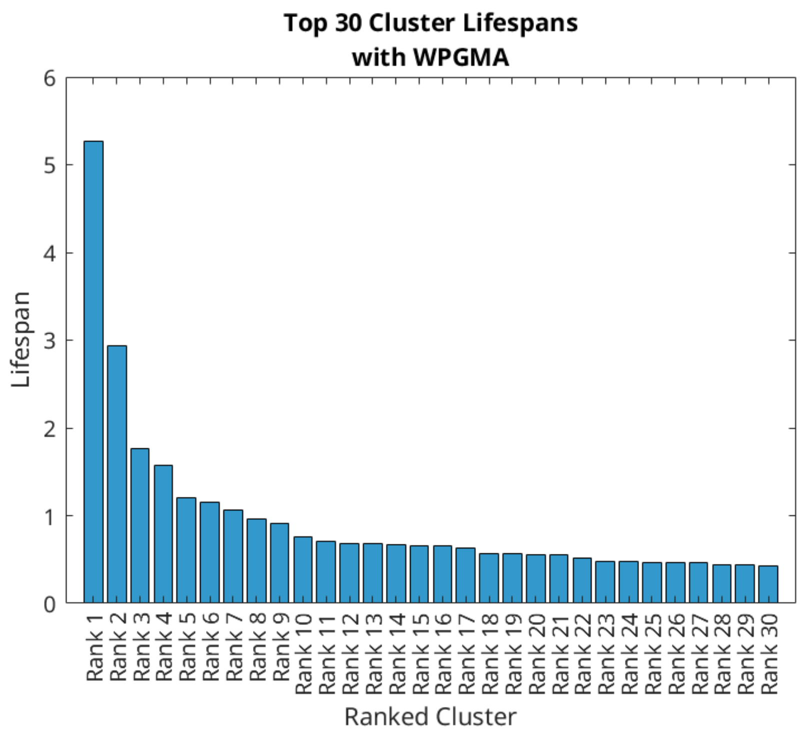
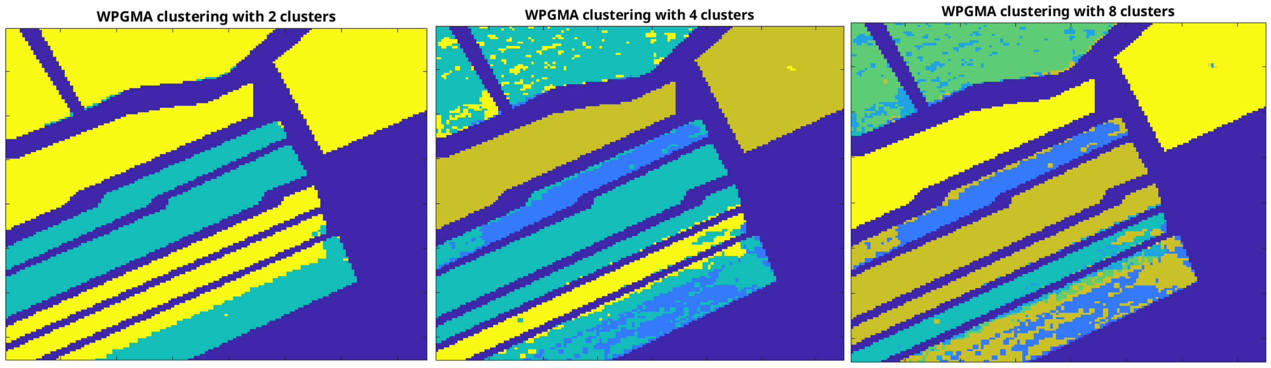


Figure 22: Lifespan of the clusters generated by the WPGMA clustering algorithm.

The WPGMA algorithm was executed for three cases, one with a cutoff at 2 clusters, one with a cutoff at 4 clusters, and a final one at 8. The algorithms for all three cases took around 2 seconds to complete, and their ARI's were 0.5891, 0.7382, and 0.8318 respectively. The figure below (Figure 23) shows the three different cutoffs.


WPGMA clustering with 2 clusters
WPGMA clustering with 4 clusters
WPGMA clustering with 8 clusters

(a) 2 clusters. (b) 4 clusters. (c) 8 clusters.

Figure 23: Comparison of the WPGMA clustering algorithm with a cutoff at 2, 4, and 8 final clusters.

3.8 Ward clustering algorithm

Finally, the same procedure was followed for the Ward algorithm. The bar plot showing the lifespans of the clusters is shown below (Figure 24). Following these findings, the number of clusters for testing in the following step was 2, 5, and 8.

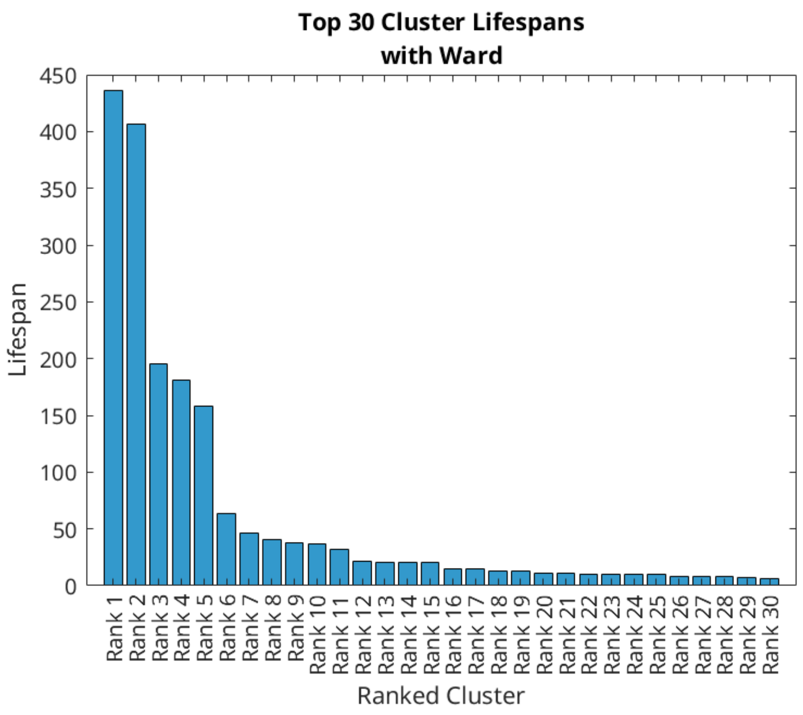
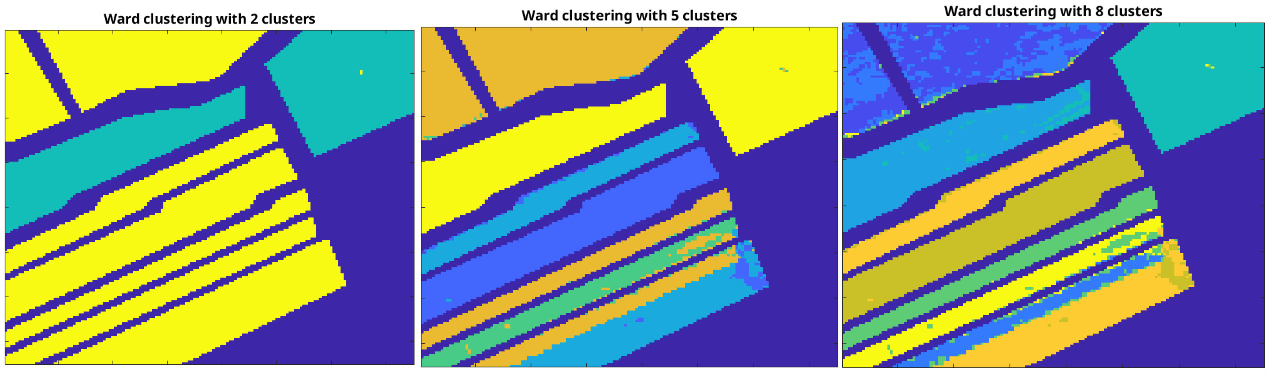


Figure 24: Lifespan of the clusters generated by the Ward clustering algorithm.

The algorithm took around 3 seconds to complete and the Adjusted Rand Indexes were 0.6032, 0.8522, and 0.9232 for the 2, 5, and 8 clusters respectively.



(a) 2 clusters. (b) 5 clusters. (c) 8 clusters.

Figure 25: Comparison of the Ward clustering algorithm with a cutoff at 2, 5, and 8 final clusters.

4 Results — Discussion

A notable characteristic of Hyperspectral Image (HSI) data is the high degree of spectral redundancy between adjacent bands. Through preprocessing, I reduced the dimensionality from 204 spectral bands to 10, achieving a 95% reduction while preserving the essential spectral information. This significant compression was made possible through spectral binning with mean aggregation, which is particularly effective for HSI data due to the high correlation between adjacent spectral bands. This correlation exists because similar materials exhibit consistent reflectance patterns across neighboring wavelengths, allowing for effective band aggregation without significant loss of discriminative information.

Analysis of the hyperspectral reflectance profiles across all pixels confirms the high correlation between adjacent wavelengths. This is illustrated in Figure 26, which shows the spectral profile of a randomly selected pixel[2]. The red dashed lines indicate wavelengths characterized by consistently low reflectance values. These absorption features can be attributed to three main factors: the intrinsic spectral properties of the target materials (vegetation in this case), atmospheric absorption bands (primarily due to water vapor and oxygen), or sensor-specific limitations in spectral sensitivity.

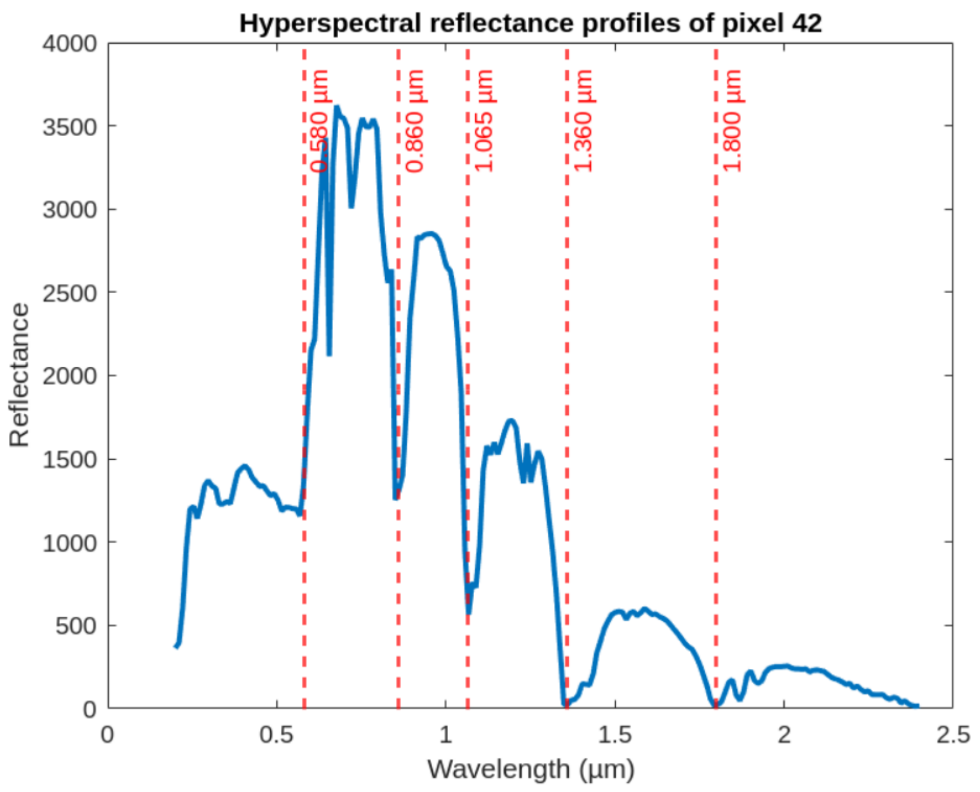


Figure 26: Hyperspectral reflectance profile of pixel 42. The dashed red lines correspond to the wavelengths that always have extremely low values.

The way the different vegetables are arranged in the Salinas field might mislead one into thinking that the clustering algorithm will have to work well with elongated clusters to correctly cluster the pixels, but in fact the clusters formed by the features are compact. This is due to the fact that the features convey only the spectral information of each data point, no spatial

[2]As if 42 is ever randomly picked.

information is found in the feature vectors; and since each type of vegetable reflects in a specific manner, then each different type must exist in compact clusters in the feature space.

One interesting modification that could be made to the features would be adding that additional spatial information about the location of each pixel (maybe by adding two more features, corresponding to the position of each pixel in the two-dimensional space). This would indeed make the clustering problem easier but it requires the knowledge that the vegetables are indeed planted in discrete plots of land, otherwise that additional information might actually make clustering the data harder.

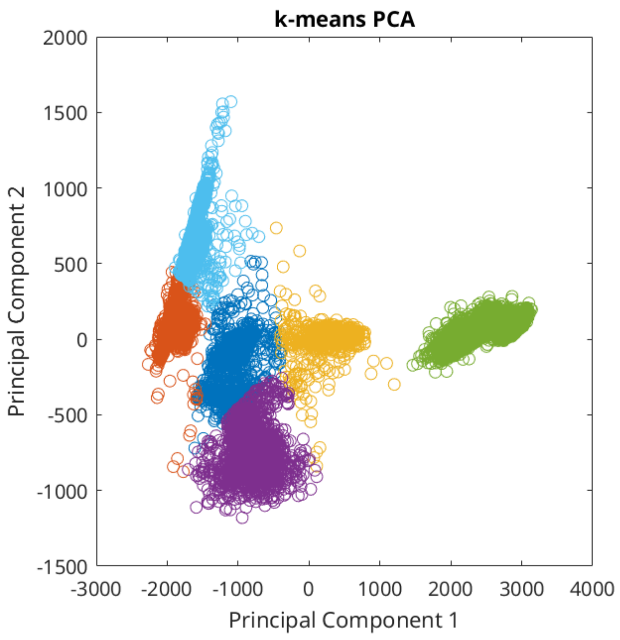
The metrics of the results of all algorithms and the several parameters tested are provided in the table below (Table 1).

Algorithm	**Parameters**	**ARI**	**RI**	**MI**	**HI**	**NMI**
k-means	$m = 5$	0.8166	0.9368	0.0632	0.8736	0.8040
	$* \ m = 8$	0.8539	0.9511	0.0489	0.9023	0.8265
	$m = 8$, no preprocess	0.8415	0.9455	0.0545	0.8910	0.8311
fuzzy c-means	$m = 8$, $q = 2$	0.9089	0.9709	0.0291	0.9417	0.8504
	$* \ m = 8$, $q = 2$, no preprocess	0.9173	0.9736	0.0264	0.9471	0.8580
possibilistic c-means	$* \ m = 8$, $q = 2$	0.7869	0.9248	0.0752	0.8497	0.7813
	$m = 8$, $q = 12$	0.6287	0.8497	0.1503	0.6993	0.6994
	$m = 8$, $q = 2$, no preprocess	0.7819	0.9230	0.0770	0.8460	0.7743
probabilistic	$* \ m = 8$	0.7719	0.9181	0.0819	0.8363	0.7374
Complete-Link	$m = 6$	0.8101	0.9348	0.0652	0.8695	0.7899
	$* \ m = 8$	0.8466	0.9486	0.0514	0.8972	0.8168
WPGMA	$m = 2$	0.5891	0.8287	0.1713	0.6575	0.9616
	$m = 4$	0.7382	0.9044	0.0956	0.8088	0.7478
	$* \ m = 8$	0.8318	0.9427	0.0573	0.8854	0.7967
$**$ Ward	$m = 2$	0.6032	0.8338	0.1662	0.6675	0.7392
	$m = 5$	0.8522	0.9494	0.0506	0.8987	0.8417
	$* \ m = 8$	0.9232	0.9756	0.0244	0.9512	0.8749

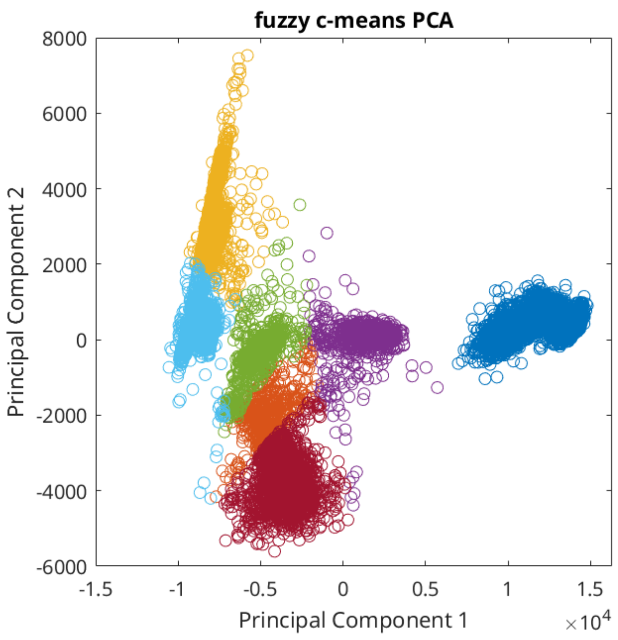
Table 1: General metrics of each clustering algorithm. For each algorithm, the parameter (or set of parameters) with the highest scores is marked by an asterisk ($*$). The algorithm with the best score is marked with two asterisks ($**$). ARI: Adjusted Rand Index, RI: Rand Index, MI: Mirkin Index, HI: Hubert Index, NMI: Normalized Mutual Information.

Of all the cost function optimization clustering algorithms, the best one that clearly outperformed the rest was the fuzzy c-means, while in the case of the hierarchical algorithms the best performing one was the Ward algorithm. Furthermore, the images generated by the cluster labels, also align with these metrics, as the highest scoring algorithms also show the data clustered better.

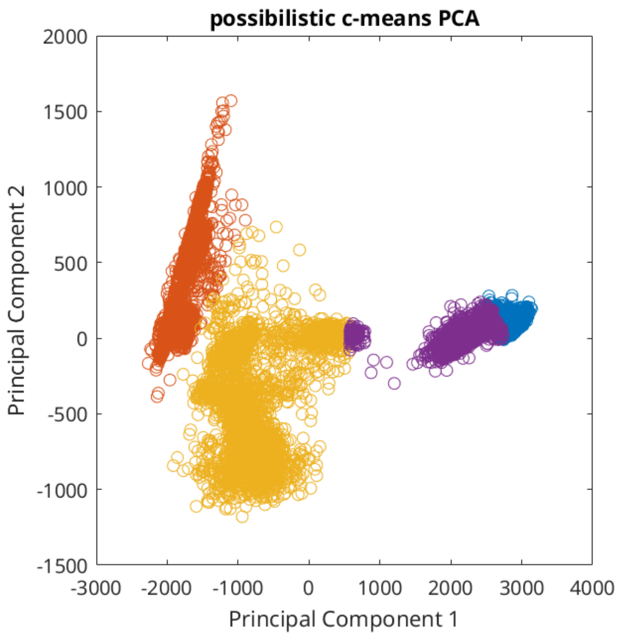
Another way to visualize how well the data were clustered is with a PCA plot of the data points colored based on the cluster labels. In Figure 27b the clustered labels are well separated, indicating that the fuzzy c-means clustering algorithm managed to separate physical clusters, while on the other hand an algorithm like the probabilistic one (Figure 27d) did not manage to separate the physical clusters that well. The possibilistic c-means algorithm reached a wrong clustering configuration, as we can see that most data points belong to one of only four clusters (Figure 27c). The k-means algorithm (Figure 27a) was also relatively accurate and that can also be seen by the well separated points on the PCA plot.



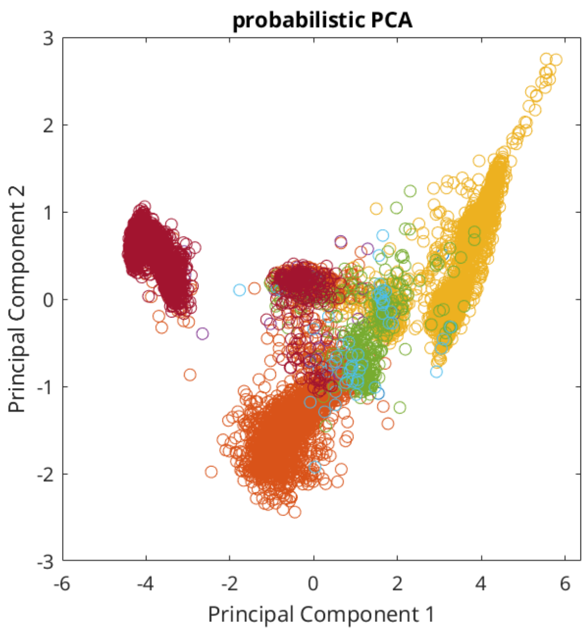
(a) k-means PCA.



(b) fuzzy c-means PCA.



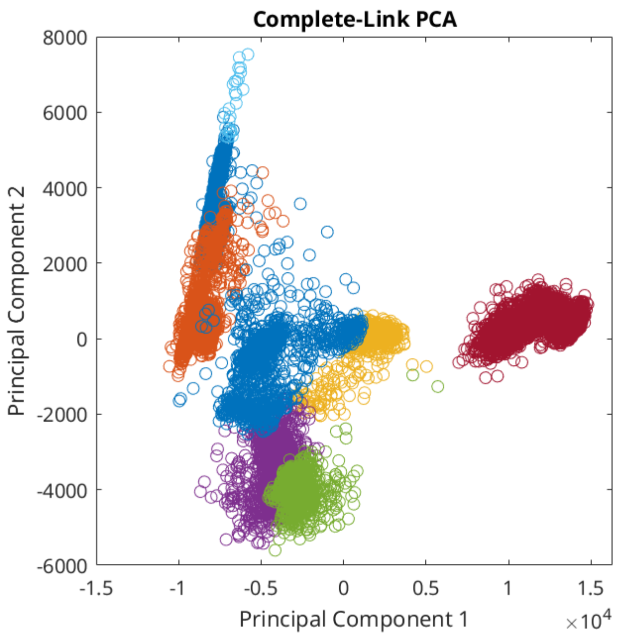
(c) possibilistic c-means PCA.



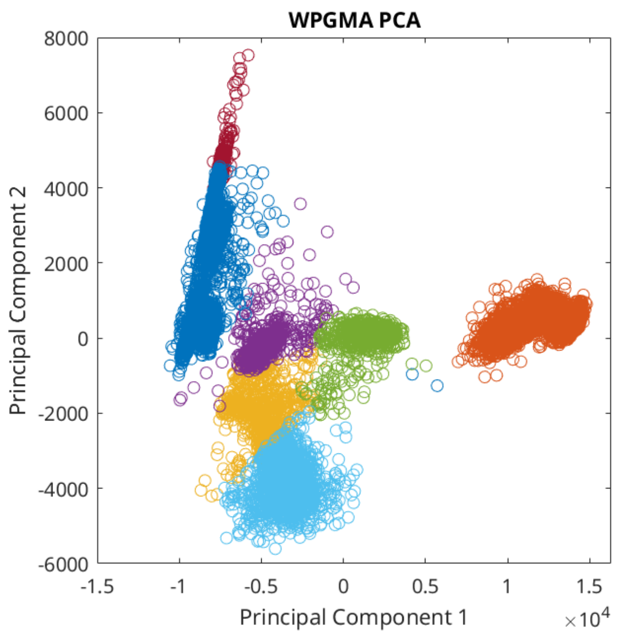
(d) probabilistic PCA.

Figure 27: PCA of the features colored by cluster label as generated by the four different cost function optimization algorithms.

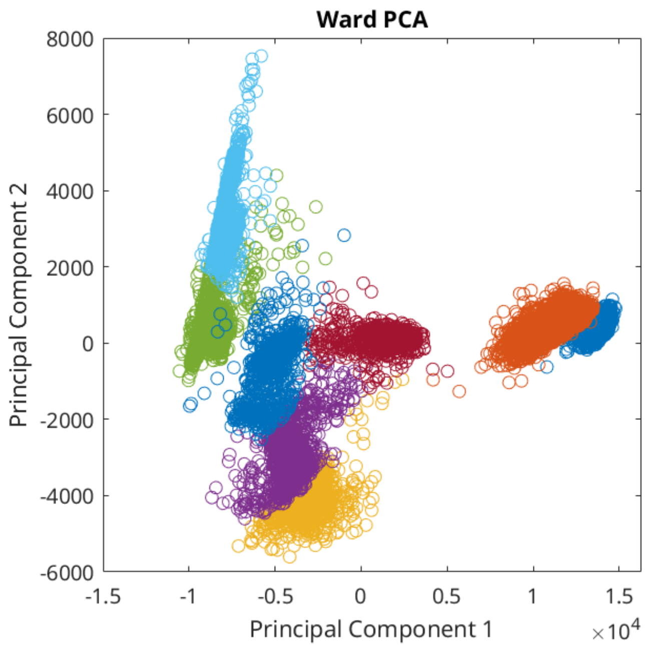
The hierarchical clustering algorithms performed over all better than the cost function optimization ones. The Complete-Link and the WPGMA algorithms performed similarly in clustering the data, as can also be seen by their PCA plots (Figures 28a and 28b). The data points as presented in the principal components space for both these algorithms are similarly separated (and dissseparated in the same positions) and their scoring metrics are also very similar. The best clustering algorithm so far though, is the Ward clustering algorithm. As is seen by Figure 28c, the data vectors are well separated and they present very low overlaps in regions of the principal components space.



(a) Complete-Link PCA.



(b) WPGMA PCA.

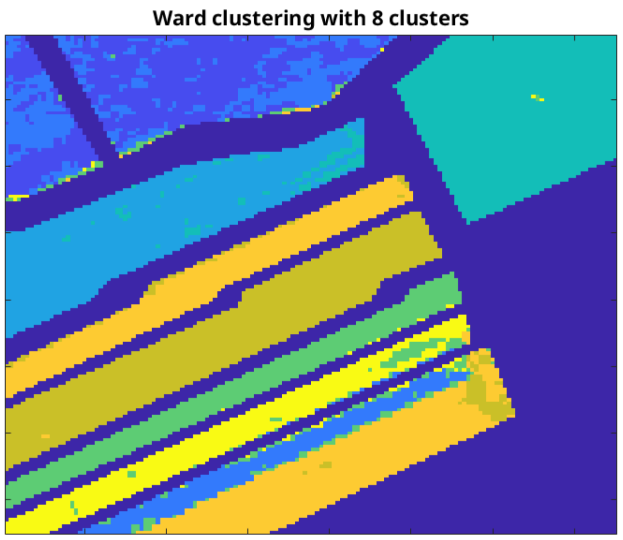


(c) Ward PCA.

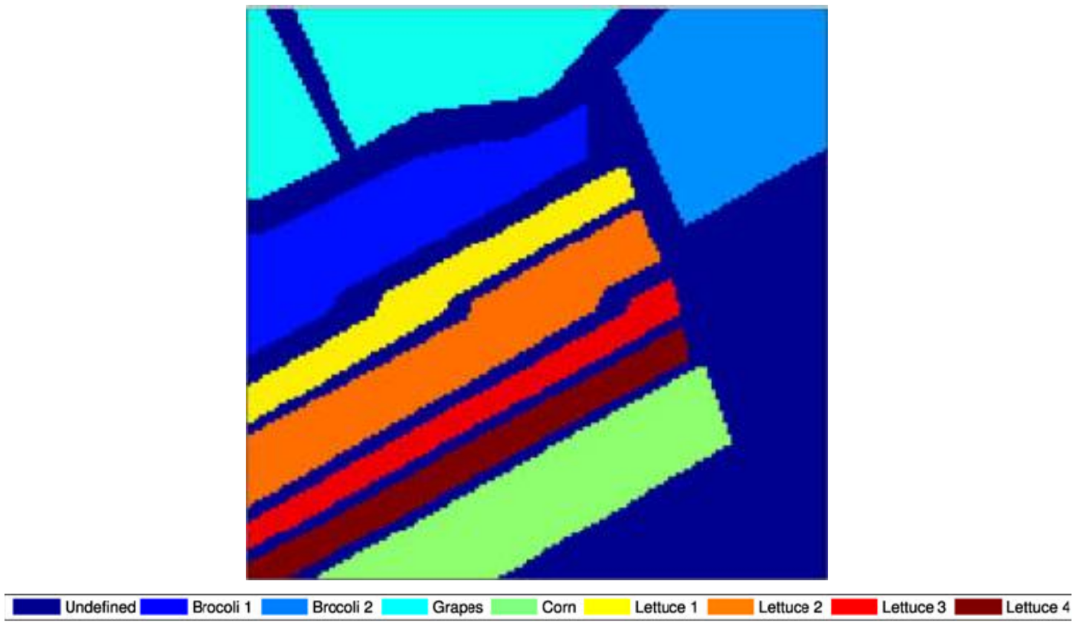
Figure 28: PCA of the features colored by cluster label as generated by the three different hierarchical algorithms.

When comparing the labels generated by the Ward algorithm with the ground truth labels, we can see that the algorithm managed to differentite almost perfectly the two types of broccoli and the four types of lettuce. It did encounter some difficulties in clustering the two plots of land that had grapes planted in them and the one plot where corn was growing. Now it is worth noting that even though some algorithms failed to cluster each different type of plant correctly, they did manage to group together same classes of types, like in the case of the Complete-Link algorithm with a cutoff at six clusters which clustered together almost perfectly the two types of broccoli (Figure 21a). Similarly, the probabilistic algorithm with $m = 8$ clusters and the Ward algorithm with a cutoff at two clusters grouped together the two types of broccoli (Figure 19 and 25a).

The fact that all algorithms, without exception, fail to cluster correctly the corn type data vectors, might even lead us to conclude that the specific plot of land with corn could have other types of vegetation growing there. The other cause of this issue is the big spatial resolution of the hyperspectral image. Since corn as a plant consists of two main colours, then it could be characterized as a hard-to-cluster plant and since the spatial resolution is $3.7m \times 3.7m$, then the pixels might not be clear enough.



(a) Cluster labels as clustered by the Ward algorithm.



(b) Ground truth labels.

Figure 29: Comparison of the labels generated by the Ward algorithm with the ground truth labels.

5 MATLAB — Appendix

All MATLAB scripts and functions used in my work, as well as the dataset which is clustered and any stored workspaces needed, are provided along with this pdf document in the same compressed folder.

The following are the required data and workspaces:

1. `data_overview.mat`
2. `Labels_Salinas.mat`
3. `preprocessed_X.mat`
4. `Salinas_Data.mat`

The following functions were created by me:

1. `calc_eta.m`
2. `corr_coeff_plot.m`
3. `eucledian_distance.m`
4. `feature_space_reduction.m`
5. `fuzzy_c_means.m`
6. `pca_plot.m`
7. `plot_lifespans.m`
8. `plotPCAClusters.m`
9. `possibi_cost.m`
10. `probabi.m`
11. `probabi_cost.m`

The following scripts and live scripts were created by me and were used in all the analyses mentioned above:

1. `data_overview.m`
2. `inspect_spectral_information.mlx`
3. `preprocess.m`
4. `results_overview.mlx`
5. `testing_CL.m`
6. `testing_fuzzy.m`
7. `testing_k_means.m`
8. `testing_possibi.m`

9. `testing_probabi.m`
10. `testing_Ward.m`
11. `testing_WPGMA.m`

The following functions were taken from the GitHub repository Introduction to Pattern Recognition a Matlab Approach or were provided in the course materials:

1. `distan.m`
2. `distant_init.m`
3. `k_means.m`
4. `pca_fun.m`
5. `possibi.m`

The following functions were taken from two GitHub repositories and were used for calculating the scoring metrics (ARI, RI, HI, MI, and NMI):

1. `Contingency.m`
   from:
   `https://github.com/areslp/matlab/blob/master/code_cospectral/Contingency.m`
2. `NMI.m`
   from:
   `https://github.com/PRML/PRMLT/blob/master/chapter01/nmi.m`
3. `RandIndex.m`
   from:
   `https://github.com/areslp/matlab/blob/master/code_cospectral/RandIndex.m`