MSc Data Science & Information Technologies

Bioinformatics - Biomedical Data Science Specialization

Course: Clustering Algorithms

Professor: Konstantinos Koutroumbas

Clustering Algorithms Final Project

Name
Konstantinos Giatras

Student ID
7115152300005

2023-2024

Contents

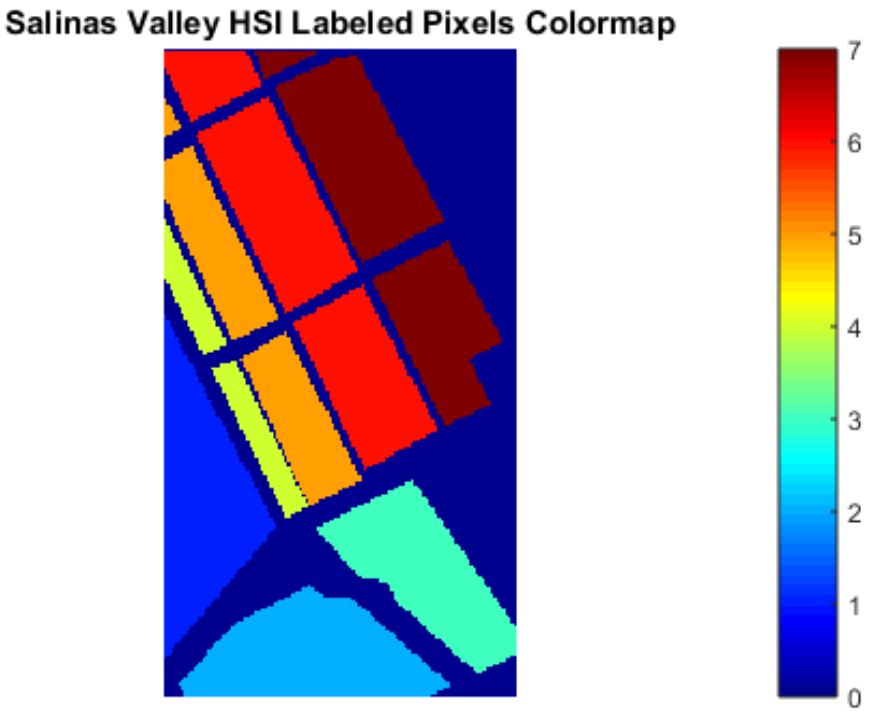
1 Introduction

This project report introduces an in-depth study on the processing of Hyperspectral Images (HSI), focusing specifically on the Salinas HSI, which represents an area in the Salinas Valley of California, USA [1]. Hyperspectral imaging is a technique that captures a scene in multiple, narrow spectral bands, creating a three-dimensional cube with two spatial dimensions and one spectral dimension. Each pixel in an HSI contains a **spectral signature** represented by a vector of these bands. In remote sensing, HSIs often contain '**mixed pixels**' where a single pixel represents multiple materials, as well as '**pure pixels**' representing a single material [5]. The project revolves around the processing of these images through clustering (unsupervised classification).

The Salinas HSI, which forms the core of this study, is a 220×120 spatial resolution image with 204 spectral bands (from 0.2μm to 2.4μm) and spatial resolution $3.7m$ (the HSI is a $220 \times 120 \times 204$ cube). Thus, a total size of $N = 26400$ sample pixels are used, stemming from **seven ground-truth classes**:

1. "grapes"
2. "broccoli"
3. "fallow 1"
4. "fallow 2"
5. "fallow 3"
6. "stubble"
7. "celery"

Note that there is no available ground truth information for the dark blue pixels (class 0) and that only the pixels with nonzero class label are taken into consideration:



The following code was used to generate the image above:

```
clear
format compact
close all
DISPLAY_FIGURES = true;

% Load the Salinas hyperspectral image (HSI) and ground truth data
load('Salinas_cube.mat')  % Load the salinas_cube matrix
load('Salinas_gt.mat')    % Load the salinas_gt matrix

% Colormap of the data based on their class labels
if DISPLAY_FIGURES
    figure;
    colormap(jet);
    imagesc(salinas_gt), axis equal
    axis off
    title('Salinas Valley HSI Labeled Pixels Colormap')
    colorbar
end
```

Notably, in clustering, the ground truth functions akin to a test set in supervised learning. Its primary role is to assess the effectiveness of various clustering techniques and to facilitate comparisons between their outcomes. However, it is important not to adjust the algorithm parameters to replicate the ground truth, as this is not the intended use of the ground truth in clustering scenarios.

This project aims to identify homogeneous regions within the Salinas HSI, using data from the $220 \times 120 \times 204$ *"**salinas_cube**"* matrix and the 220×120 *"**salinas_gt**"* class label image. The methodologies employed include various Cost-Function Optimization (CFO) and Hierarchical clustering algorithms:

- CFO: **k-means**, **fuzzy c-means**, **possibilistic c-means**, **probabilistic c-means** (where each cluster is modelled by a normal distribution)
- Hierarchical: **Complete-Link**, **WPGMC**, **Ward's algorithm**

The performance of these algorithms will be assessed in finding homogeneous regions, accounting for the class label information and analyzing the principal components of the image. This report will detail both qualitative and quantitative assessments of these algorithms, discussing their relative efficacies and any notable challenges encountered during the process.

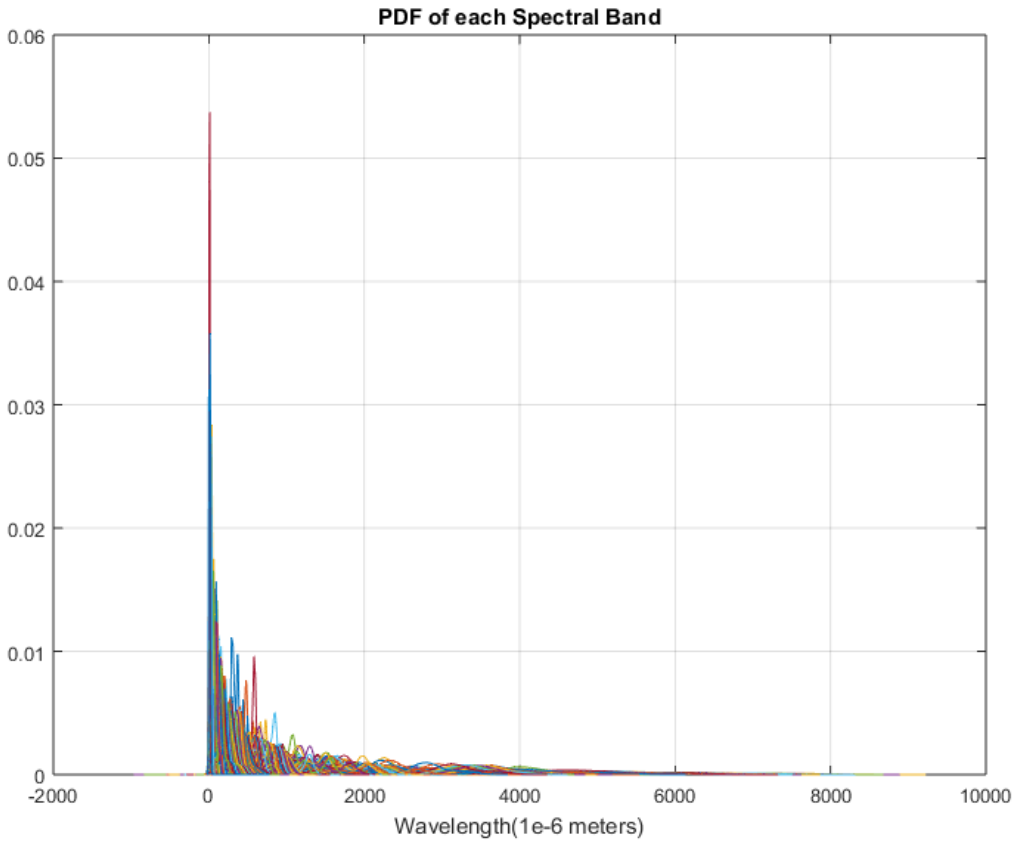
2 "Feeling the Data"

We begin by modifying the Salinas cube matrix for clustering. This involves transforming it into a 3-dimensional matrix. In this new matrix, each row represents a pixel, and the columns indicate the spectral values over 204 bands. We also exclude pixels associated with the 0 class, as these lack corresponding ground truth data. The following code updates the dataset dimensions, reshapes it for processing, filters out the pixels without class labels, and defines the new ground truth classes:

```
% The Salinas HSI is currently in a 220x120x204 cube
[p, n, l] = size(salinas_cube);  % p and n are spatial dimensions, l is spectral

% Reshape the HSI cube for clustering
X_total = reshape(salinas_cube, p*n, l);

% Adjust the ground truth labels
L = reshape(salinas_gt, p*n, 1);

% Filter out pixels with no class label (class label 0)
existed_L = (L > 0);
X = X_total(existed_L, :);
[px, nx] = size(X);

% Update the ground truth to consider only existing labels
Ground_truth = L(existed_L);

% Define the new class labels for Salinas HSI
Labels = {1, 'Grapes'; 2, 'Broccoli'; 3, 'Fallow 1'; 4, 'Fallow 2'; 5, 'Fallow 3'; 6, '
    Stubble'; 7, 'Celery'};
Labels_table = cell2table(Labels, 'VariableNames', {'Class_Number', 'Class_Label'});
```

2.1 Descriptive Statistics

After reshaping the HSI cube into a 3-D matrix, we use the following code to plot the probability density funtion (pdf) of each spectral band:



```
% Plot PDF of each spectral band
if DISPLAY_FIGURES
    figure;
    for i = 1:204
        ksdensity(X(:,i))
        hold on
        grid on
    end
    xlabel('Wavelength(1e-6 meters)')
    title('PDF of each Spectral Band')
    hold off
end
```

From the approximation of the PDFs, we can observe that the largest density of values can be found in the visible spectrum (380-700 nm) [2], with a few peaks in the infrared spectrum. In addition, most of the values range between 0 and 2000 μm.

We use the following code to extract some more information about the data:

```
% Find descriptive statistics about the dataset
max_X = max(max(X));
min_X = min(min(X));
mean_array = mean(X);
std_X = std(mean_array);
```

The maximum value of a spectral band is 8117, while the minimum is -11. Also,the mean values of the spectral bands are spread out, with a standard deviation of 1187.75014797152.

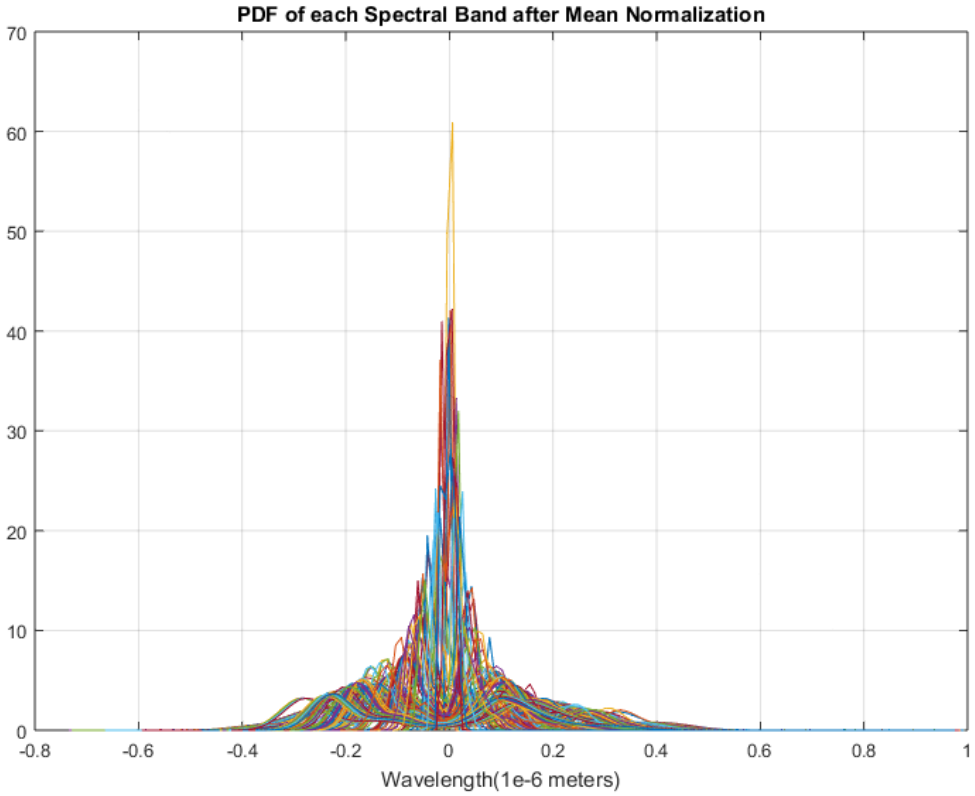
2.2 Mean Normalization

Given the wide range of values in the spectral bands, and considering our objective of assessing various clustering techniques, it's recommended to normalize the data. This is particularly important for distance-based clustering methods like k-means, which can be affected by larger distances and may take longer to converge as a result. We'll be using mean normalization, which is calculated using the formula:

$$\frac{x - \bar{x}}{max(x) - min(x)}$$

We use the following code:

```
% Calculate mean, max, and min for each column (spectral band)
mean_X_total = mean(X_total, 1);
max_X_total = max(X_total, [], 1);
min_X_total = min(X_total, [], 1);

% Normalize X_total using bsxfun for correct dimension matching
X_total_norm = bsxfun(@minus, X_total, mean_X_total);
X_total_norm = bsxfun(@rdivide, X_total_norm, (max_X_total - min_X_total));

% Extract the normalized data for existing labels
X = X_total_norm(existed_L, :);

% Plot PDF of each spectral band after mean normalization
if DISPLAY_FIGURES
    figure;
    for i = 1:204
        ksdensity(X(:,i))
        hold on
        grid on
    end
    xlabel('Wavelength(1e-6 meters)')
    title('PDF of each Spectral Band after Mean Normalization')
    hold off
end
```



This plot suggests that the spectral bands have been successfully centralized around zero, with a consistent range across bands to prevent scale-induced bias in clustering. However, pronounced peaks near zero raise the possibility of outliers or narrow distributions, which may require additional scrutiny due to their potential influence on clustering results.

It is important to note that, although mean normalization adjusts the position and scale of the data, it does not alter its original distribution. Thus, algorithms reliant on normality, such as probabilistic c-means may necessitate further standardization processes.

2.3 Principal Component Analysis (PCA)

Implementing Principal Component Analysis (PCA) following mean normalization is a strategic approach in hyperspectral image analysis, particularly when dealing with the high dimensionality and potential presence of outliers or skewed distributions. PCA serves to emphasize the most significant features of the data by transforming it into a set of principal components that capture the majority of its variance in a reduced-dimensional space. This transformation not only aids in outlier detection by revealing data points that deviate from the main variance direction but also reduces noise, smoothing out any effects from narrow distributions. While PCA does not remove outliers, it repositions them to reduce their influence on the analysis. The 'explain' variable in PCA provides valuable insights into the proportion of variance each principal component captures, guiding the selection of components for effective data representation. [3]

We use the following code:

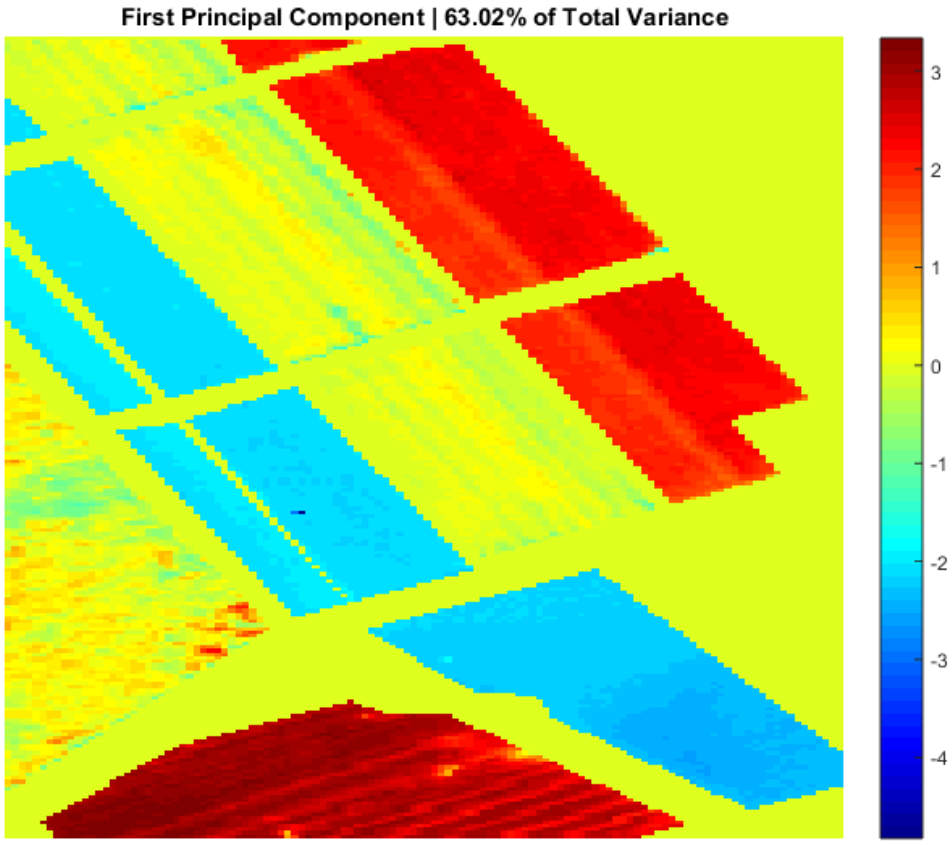
```
% Perform PCA on the normalized data using pca_fun
[eigenval, eigenvec, explain, Y, mean_vec] = pca_fun(X', 3);

% Extract the scores (the projections onto the principal components)
Z = Y';  % Transpose to match the pca output format

% Calculate the cumulative variance explained by the first three components
explained_variance = sum(explain(1:3));

% Placeholder for the first principal component with the original image dimensions
```

```
PC1_reshaped = zeros(p*n, 1);

% Fill in the values for the non-zero labeled pixels
PC1_reshaped(existed_L) = Z(:,1);

% Reshape the placeholder to the original spatial dimensions
PC1_reshaped = reshape(PC1_reshaped, p, n);

% Display the variance explained by the first three principal components
disp('Variance explained by the first three principal components:');
disp(array2table(explain(1:3)', 'VariableNames', {'First_PC', 'Second_PC', 'Third_PC'}))
    ;
```

From output of the above, we can see that the first principal component (PC) explains 63.015% of the variance in the dataset, the second explains 34.634%, and the third explains 1.2379%. This is a typical output for PCA, where the first few PCs explain the majority (∼100% in this case) of the variance, and subsequent components explain progressively less. This is a strong indication that these first three PCs capture the most critical information from the original dataset.

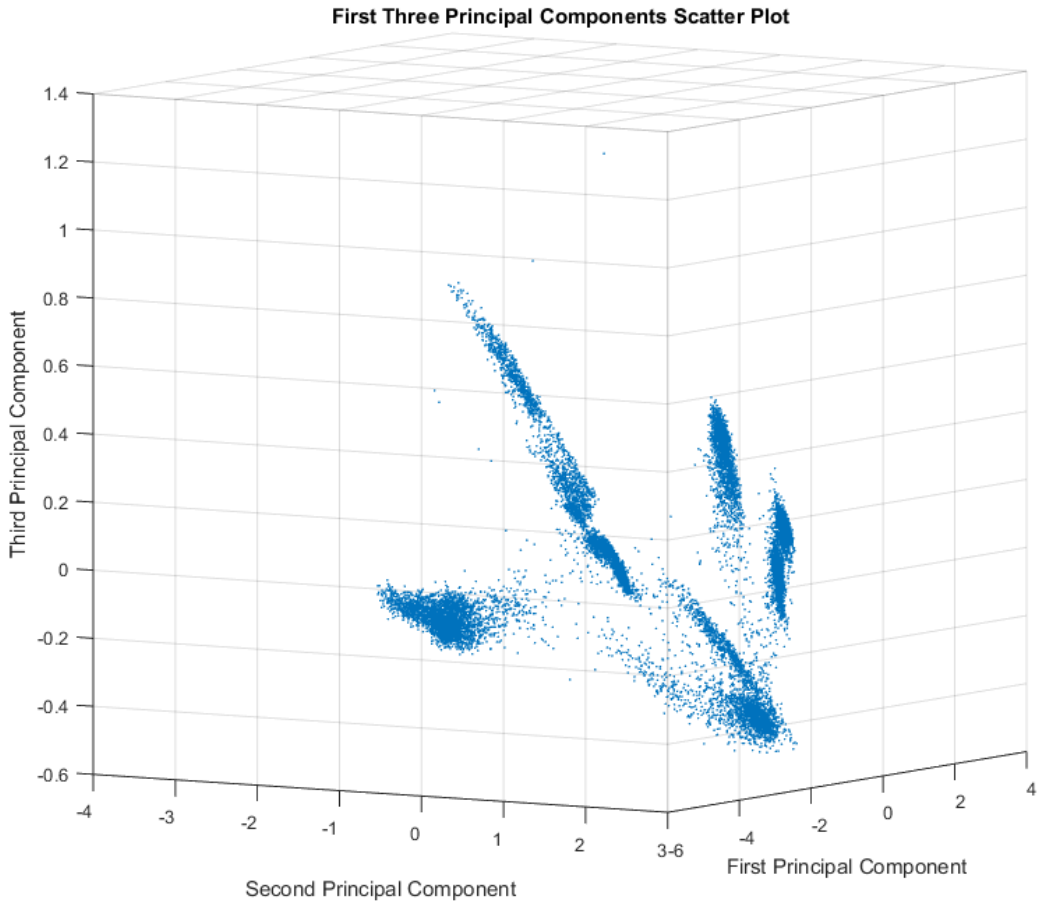
In order to have a visual reference of the clusters in our dataset, we plot the corresponding depiction of the first PC, which captures the majority of the total variance (63.02%):

```
% Plot the first principal component
if DISPLAY_FIGURES
    figure;
    imagesc(PC1_reshaped);
    colormap(jet);
    colorbar;
    title(sprintf('First Principal Component | %.2f%% of Total Variance', explain(1) *
    100));
    axis off;
end
```



We can also display the projected pixels in a 3D space defined by the first three principal components. It's important to note that the clusters formed here may not be the same as those in the original 204-dimensional space. This difference occurs because keeping the variance intact doesn't always mean that the original clusters will be maintained. However, it's interesting to observe that about 6 to 8 clusters emerge in this representation:

```
% Scatter plot of the first three principal components
if DISPLAY_FIGURES
    figure;
    scatter3(Z(:,1), Z(:,2), Z(:,3), 10, '.');  % The '10' is the size of the markers
    title('First Three Principal Components Scatter Plot');
    xlabel('First Principal Component');
    ylabel('Second Principal Component');
    zlabel('Third Principal Component');
    grid on;
    view([58 -6]);
end
```



Note that the specific implementation of the *pca_fun* function used here was taken from (c) 2010 S. Theodoridis, A. Pikrakis, K. Koutroumbas, D. Cavouras.

3 Identification of the Number of Physical Clusters

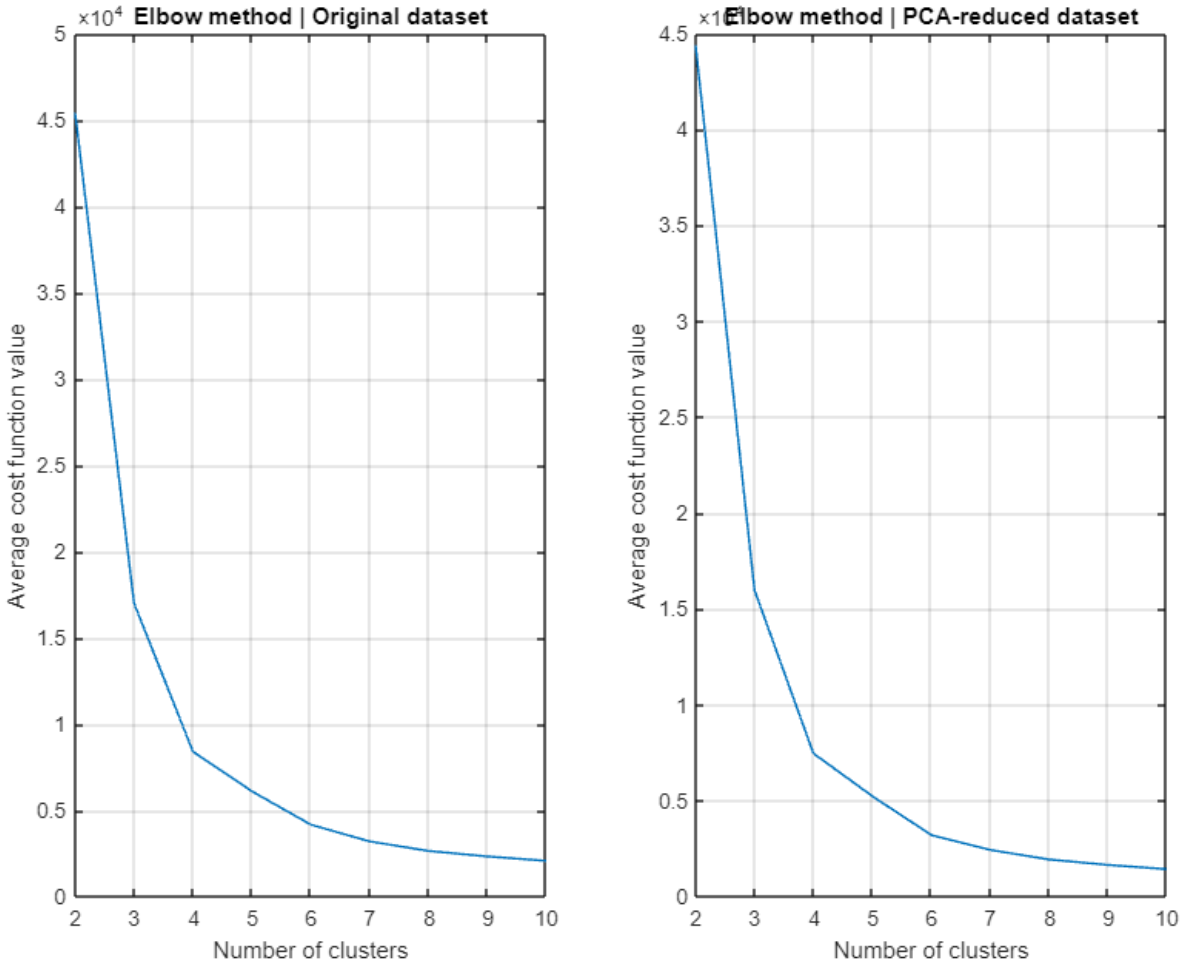
3.1 Elbow Method K-Means

First, we implement the k-means clustering algorithm (specifically taken from (c) 2010 S. Theodoridis, A. Pikrakis, K. Koutroumbas, D. Cavouras), applying the elbow method to both the original 204-dimensional dataset and the 3-dimensional dataset obtained from PCA.

The elbow method is a technique used to find the ideal number of clusters in k-means clustering. This method is grounded in the observation that the cost function of k-means, denoted as $J(U, \Theta) = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij} ||x_i - \theta_j||^2$, tends to decrease as the number of clusters increases. This decrease occurs because more clusters mean more centroids for data points to be assigned to, thereby reducing their average distance to these centroids, which in turn lowers the cost function value. However, beyond a certain point, adding more clusters doesn't significantly reduce the cost function value. This happens because new centroids will likely fall into existing clusters, marginally decreasing the distances within clusters.

In our approach, we conduct 40 trials of the k-means algorithm, each with 10 iterations, varying the number of clusters from 2 to 10. We then calculate the average cost function value for each cluster number across these trials and plot these averages against the respective cluster numbers:

```
% Initialize arrays to store the sum of within-cluster distances
cost_stacked_total = [];
cost_stacked_PCA = [];

% Perform k-means clustering on the original dataset for a range of cluster numbers
for i = 1:40
    cost_total = [];
    for k = 2:10
        [idx, ~, sumd] = kmeans(X, k, 'Display', 'final', 'MaxIter', 1000, 'Replicates', 5);
        cost_total = cat(1, cost_total, sum(sumd));
    end
    cost_stacked_total = cat(2, cost_stacked_total, cost_total);
end

% Perform k-means clustering on the PCA reduced dataset for a range of cluster numbers
for i = 1:40
    cost_total = [];
    for k = 2:10
        [idx, ~, sumd] = kmeans(Z, k, 'Display', 'final', 'MaxIter', 1000, 'Replicates', 5);
        cost_total = cat(1, cost_total, sum(sumd));
    end
    cost_stacked_PCA = cat(2, cost_stacked_PCA, cost_total);
end

% Calculate the mean cost for each number of clusters
mean_cost_stacked_total = mean(cost_stacked_total, 2);
mean_cost_stacked_PCA = mean(cost_stacked_PCA, 2);

% Plot the results
if DISPLAY_FIGURES
    figure;
    subplot(1,2,1);
    plot(2:10, mean_cost_stacked_total);
    title('Elbow method | Original dataset')
    xlabel('Number of clusters')
    ylabel('Average cost function value')
    xticks(2:10)
    grid on;

    subplot(1,2,2);
    plot(2:10, mean_cost_stacked_PCA)
    title('Elbow method | PCA-reduced dataset')
    xlabel('Number of clusters')
    ylabel('Average cost function value')
    xticks(2:10)
    grid on;
end
```



The graph shows that the best number of clusters for both the original 204-dimensional dataset and the 3-dimensional dataset obtained through PCA lies between 4 and 6 (with a prominent knee observed for 4 clusters). Despite a reduced rate of decrease in the cost function value after three clusters, the reduction remains noteworthy. Consequently, the solution with three clusters isn't considered the best.

3.2 Silhouette Method K-Means

Now that we have a basic understanding of the ideal number of clusters, we can increase our certainty through additional evaluation. An effective technique for this is the silhouette method. This method assesses the quality of clustering by determining how cohesive the clusters are. It involves calculating the silhouette coefficient, which has a value between -1 and 1. [4] Here's what these values mean:

- A coefficient close to 1 signifies that the clusters are very cohesive
- A coefficient near 0 suggests that some points of a cluster are close to the border of another cluster
- Negative values indicate overlapping between clusters

The silhouette coefficient considers two factors, cluster cohesion and cluster separation:

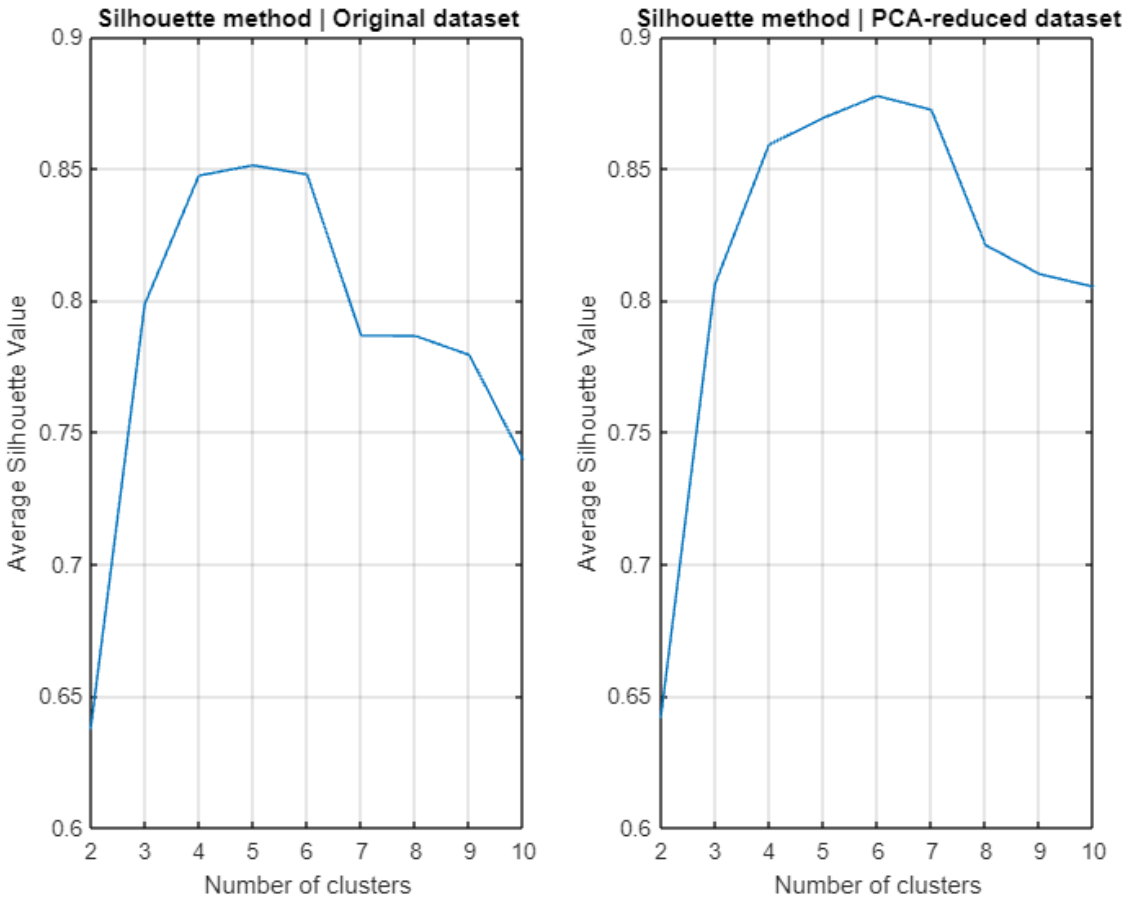
1. Cohesion refers to the average distance between a data point and all other points in the same cluster. For each data point x in a cluster C_i, cohesion is calculated as $a(x) = \frac{\sum_{x \in C_i} d(x, x \notin C_i)}{|C_i| - 1}$.
2. Separation measures the average distance between a data point and all points in the nearest cluster. For a data point x in cluster C_j, separation is $b(x) = min \frac{\sum_{x \in C_j} d(x, x_{\in C_j})}{|C_j|}$.

The overall silhouette coefficient for a set of clusters is calculated by taking the average of the difference between separation and cohesion, divided by the greater of the two, for all clustered instances x:

$$s = \frac{1}{N} \sum_{i}^{n} \frac{b(i) - a(i)}{max(a(i), b(i))}$$

Using this silhouette method, we can apply the MATLAB evalclusters function to determine the average silhouette coefficient for each cluster number. This can be done for both the original 204-dimensional dataset and the dataset reduced to 3 dimensions using PCA, using the k-means clustering algorithm:

```
% Perform silhouette analysis on the original normalized dataset
evaluation_total = evalclusters(X, 'kmeans', 'silhouette', 'KList', 1:10);

% Perform silhouette analysis on the PCA-reduced dataset
evaluation_PCA = evalclusters(Z, 'kmeans', 'silhouette', 'KList', 1:10);

% Plot the silhouette analysis results
if DISPLAY_FIGURES
    figure;

    % Silhouette analysis for the original normalized dataset
    subplot(1,2,1);
    plot(evaluation_total.CriterionValues);
    title("Silhouette method | Original dataset")
    xlabel('Number of clusters')
    ylabel('Average Silhouette Value')
    xticks(1:10);
    grid on;

    % Silhouette analysis for the PCA-reduced dataset
    subplot(1,2,2);
    plot(evaluation_PCA.CriterionValues);
    title("Silhouette method | PCA-reduced dataset")
    xlabel('Number of clusters')
    ylabel('Average Silhouette Value')
    xticks(1:10);
    grid on;
end
```



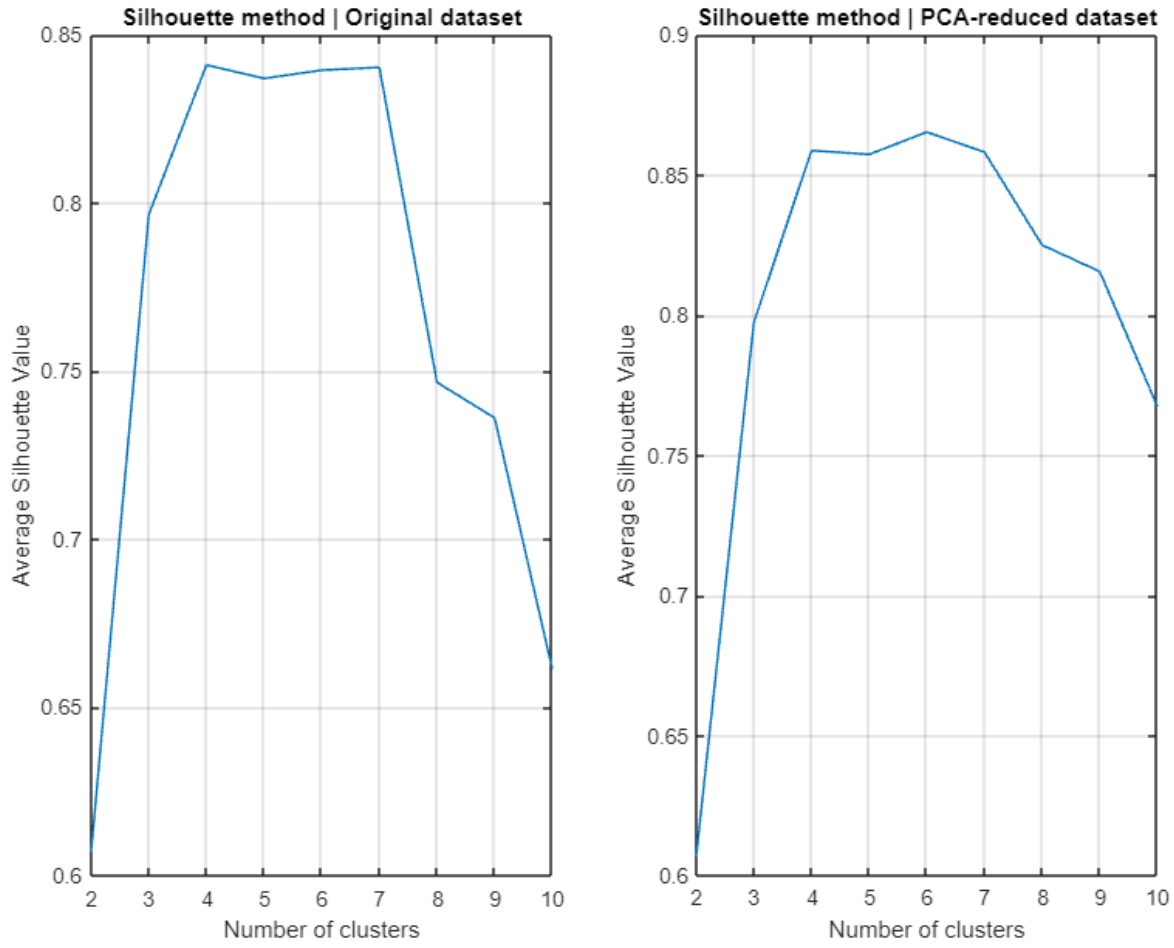
Just like earlier, we are not considering the solution with 3 clusters as the best choice, for the reasons already discussed. The results from evaluating clustering, whether using the original dataset or the projected one, seem to lead to the same conclusions. Given these findings, the best number of clusters for our purpose of pinpointing uniform areas in the Salinas Valley is 6, because it is the highest number of clusters that still shows the best performance.

3.3 Silhouette Method Hierarchical

Finally, in order to account for the potential non-compact cluster shapes, we will run the same evaluation as before using a hierarchical framework (utilizing the '*linkage*' MATLAB command):

```
% Perform silhouette analysis using linkage on the original normalized dataset
evaluation_total = evalclusters(X, 'linkage', 'silhouette', 'KList', 1:10);

% Perform silhouette analysis using linkage on the PCA-reduced dataset
evaluation_PCA = evalclusters(Z, 'linkage', 'silhouette', 'KList', 1:10);

% Plot the silhouette analysis results
if DISPLAY_FIGURES
    figure;

    % Silhouette analysis for the original normalized dataset
    subplot(1,2,1);
    plot(evaluation_total.CriterionValues);
    title('Silhouette method | Original dataset')
    xlabel('Number of clusters')
    ylabel('Average Silhouette Value')
    xticks(1:10);
    grid on;

    % Silhouette analysis for the PCA-reduced dataset
    subplot(1,2,2);
    plot(evaluation_PCA.CriterionValues);
```

```
    title('Silhouette method | PCA-reduced dataset')
    xlabel('Number of clusters')
    ylabel('Average Silhouette Value')
    xticks(1:10);
    grid on;
end
```



Based on our analysis, it seems that choosing 4 clusters might be ideal for identifying uniform areas in the Salinas Valley. Nonetheless, the option with 6 clusters also shows a commendably high Silhouette score, and this count is the maximum before we observe a notable decline in performance.

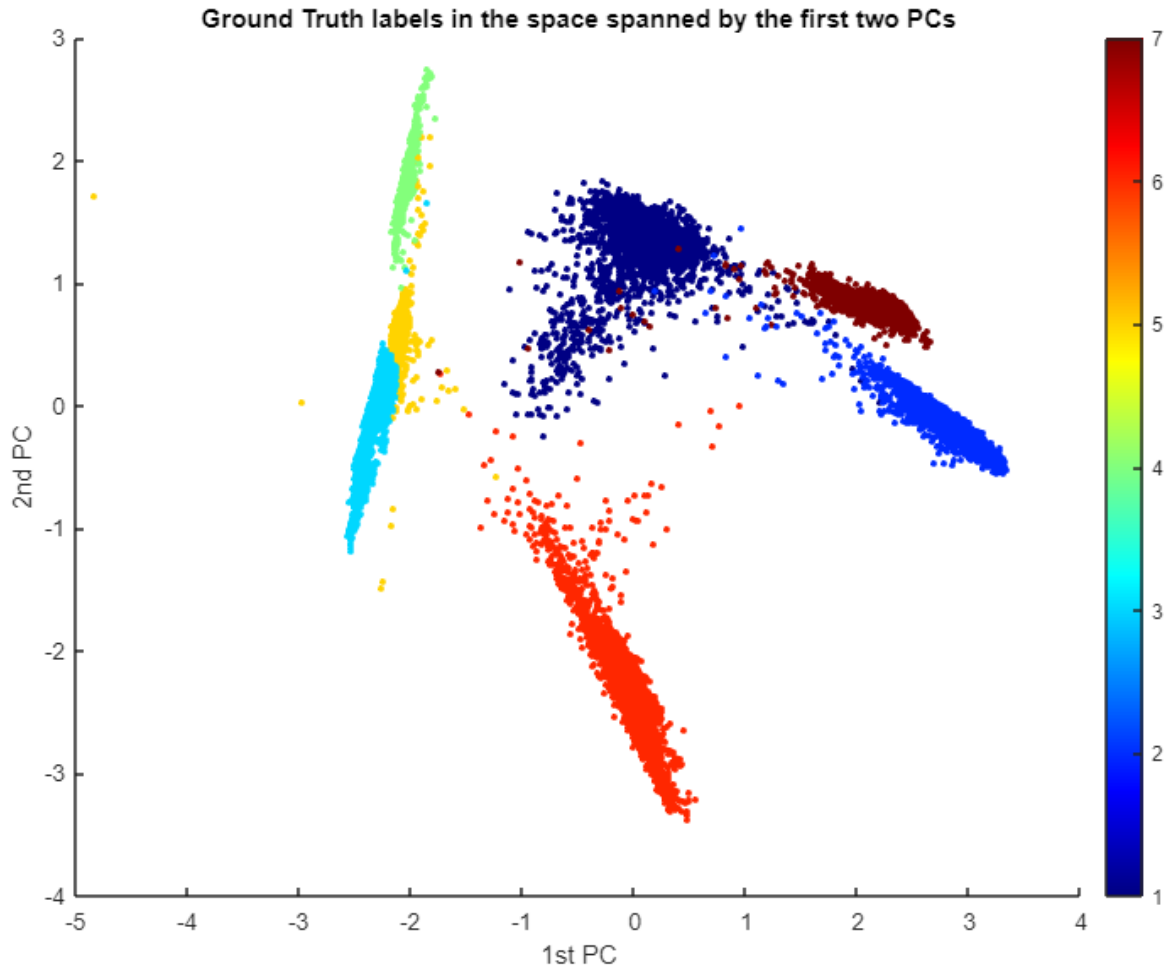
In conclusion, we determine that **6** is the **best number of clusters** for this task. This is mainly because 6 is the last count where we see a significant decrease in the rate of decline of the cost function using the elbow method, and there is also a noticeable drop in Silhouette values for any higher number of clusters.

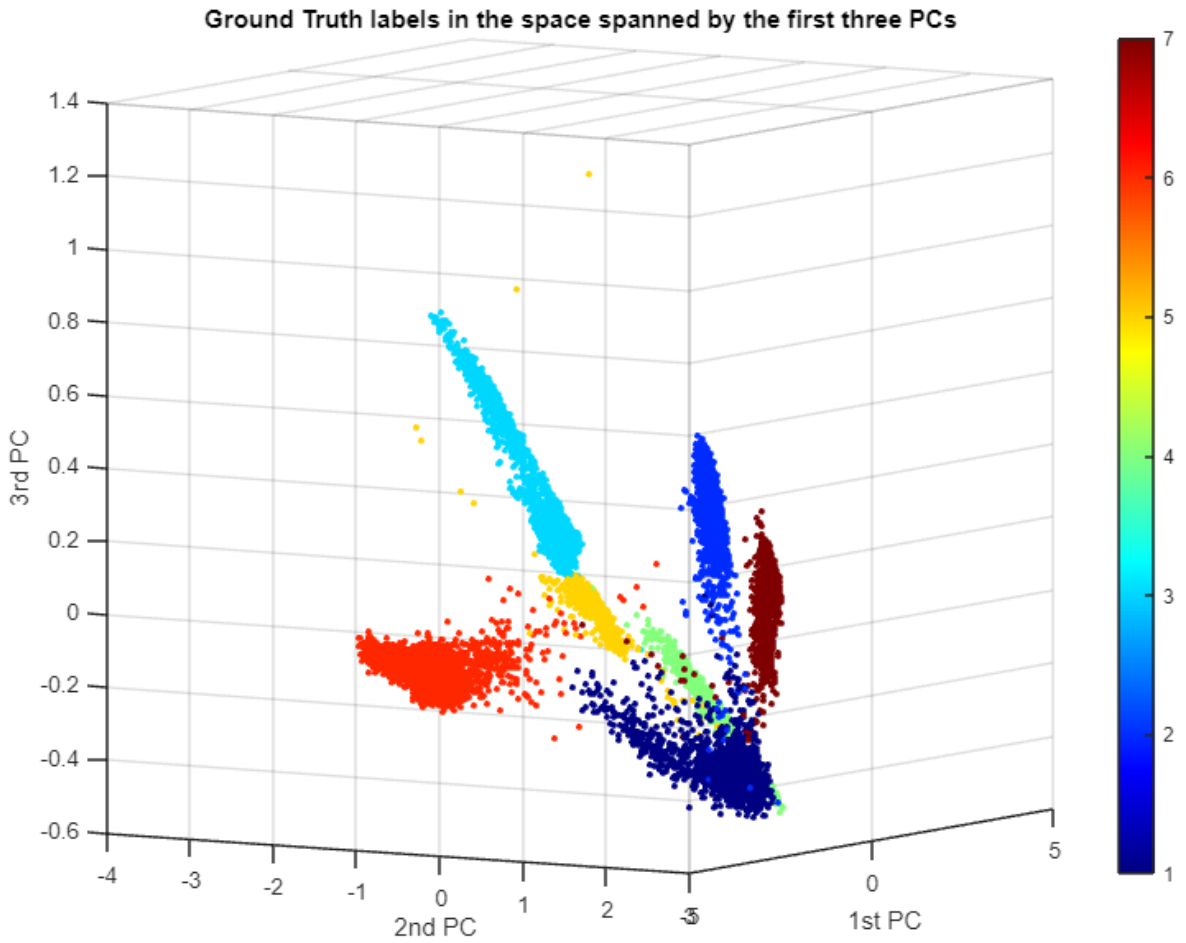
Furthermore, the 3-dimensional dataset, which we derived from applying PCA to the original 204-dimensional dataset, appears to preserve the physical clusters. This preservation is evident as we see similar results in cluster evaluations using both the 3-dimensional and the original 204-dimensional datasets. To maintain the effectiveness of the Euclidean distance, which becomes less effective in high dimensional spaces, while also enhancing computational efficiency, we will proceed by running the algorithms in the space defined by the first three principal components of the original dataset.

4 Execution of the Algorithms and Qualitative Evaluation

To better understand how the ground truth classes are distributed in the reduced-dimensional space created by PCA, we plot the ground truth labels using the first two principal components and then with the first three principal components:

```
% Plot ground truth labels in the space of the first two principal components
if DISPLAY_FIGURES
    figure;
    scatter(Z(:,1), Z(:,2), 8, Ground_truth, 'filled', 'MarkerEdgeAlpha', 0.9); % The
    '36' is the size of the markers
    title('Ground Truth labels in the space spanned by the first two PCs')
    xlabel('1st PC')
    ylabel('2nd PC')
    colormap('jet')
    colorbar
end

% Plot ground truth labels in the space of the first three principal components
if DISPLAY_FIGURES
    figure;
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Ground_truth, 'filled'); % The '36' is the size
    of the markers
    title('Ground Truth labels in the space spanned by the first three PCs')
    xlabel('1st PC')
    ylabel('2nd PC')
    zlabel('3rd PC')
    colormap('jet')
    colorbar
    view([58 -6])
end
```





We observe that the addition of the third principal component axis, which accounts for only 1.237% of the total variance, results in a better distinction between the physical clusters. This suggests that any clusters beyond the previously computed optimal number of 6 might be composed of spectral band values that are very similar to each other.

To assess how well the various clustering algorithms perform, we ran each algorithm for two scenarios:

1. For **six clusters**, determined as the optimal number from our previous analysis
2. For **seven clusters**, which corresponds to the ground truth

4.1 Cost Function Optimization (CFO) Clustering Algorithms

For the Cost Function Optimization (CFO) algorithms, we executed each algorithm with random seed values ranging from 1 to 15. We then selected the best performance based on the average silhouette coefficient value. This approach helps us eliminate runs with poor outcomes due to suboptimal initial conditions. Examples of such conditions include poor placement of starting points in algorithms like k-means and fuzzy c-means, or subpar local optima in the Expectation Maximization (EM) algorithm for probabilistic clustering.

4.1.1 K-Means

The k-means algorithm is a type of hard clustering method where each data point is exclusively assigned to a single cluster. It starts by choosing initial cluster centers at random within the dataset. Then, every data point is associated with the nearest center, and these centers are adjusted to be at the average location of the points assigned to them. This procedure is repeated over and over until there is no significant reduction in the cost function $J(U, \Theta) = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij} ||x_i - \theta_j||^2$, signaling that the algorithm has converged. Here, the specific implementation of the k-means algorithm was taken from (c) 2010 S. Theodoridis, A. Pikrakis, K. Koutroumbas, D. Cavouras.

We used the following code:

```
% Initialize variables to store the best silhouette scores and corresponding seeds
Best_silhouette_score_kmeans = zeros(1, 15);  % One entry for each seed
Best_seed = 0;
numClustersOptions = [6, 7];
Best_clustering_indices = cell(1, length(numClustersOptions));  % Store clustering results for each k


% Iterate over seed values
for seed = 1:15
    rng(seed, 'twister'); % Set the seed for reproducibility
    temp_clustering_indices = cell(1, length(numClustersOptions));  % Temporary storage for clustering indices
    silhouette_scores = zeros(1, length(numClustersOptions));  % Store silhouette scores for each k

    % Perform k-means clustering and silhouette analysis for each number of clusters
    for clusterOption = 1:length(numClustersOptions)
        currentNumClusters = numClustersOptions(clusterOption);
        idx = kmeans(Z, currentNumClusters, 'Distance', 'sqeuclidean');
        temp_clustering_indices{clusterOption} = idx;  % Store the clustering indices
        s = silhouette(Z, idx, 'sqeuclidean');
        silhouette_scores(clusterOption) = mean(s);
    end

    % Compute the average silhouette score for the current seed across both k=6 and k=7
    average_silhouette_score = mean(silhouette_scores);

    % Store the seed, its score, and clustering indices if it is the best so far
    if average_silhouette_score > max(Best_silhouette_score_kmeans)
        Best_silhouette_score_kmeans = silhouette_scores;
        Best_seed = seed;
        Best_clustering_indices = temp_clustering_indices;  % Store the best clustering indices
    end
end

% Display the best overall seed and the corresponding silhouette scores for each k
disp(['Best overall seed: ' num2str(Best_seed)]);
disp('Silhouette scores for the best seed:');
disp(array2table(Best_silhouette_score_kmeans, 'VariableNames', {'k6', 'k7'}));
```

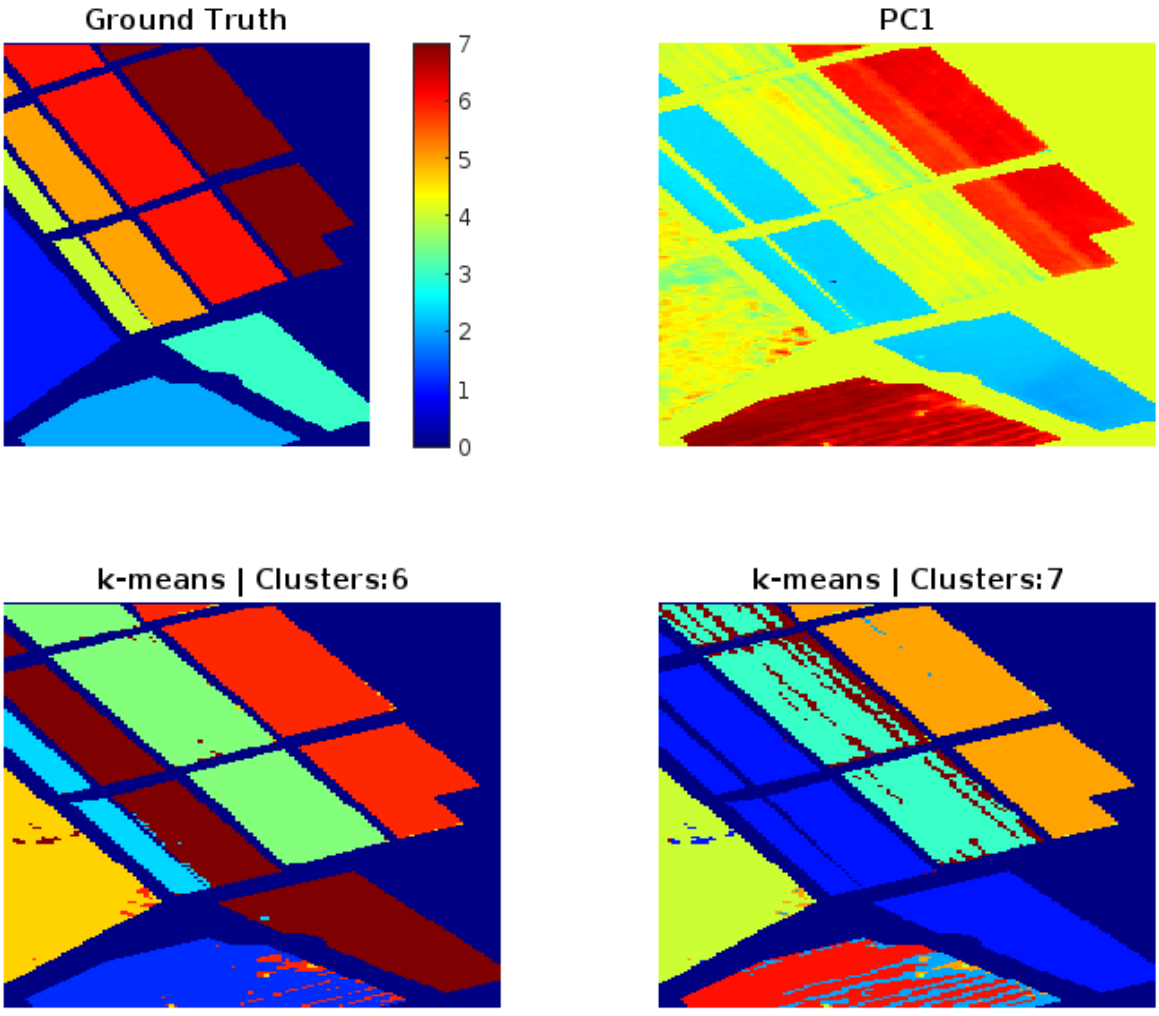
The best performance of the k-means algorithm, after iterating through 15 seeds, was obtained from random seed 2, with Silhouette scores: 0.87751 (for the 6 clusters case) and 0.77807 (for the 7 clusters case).

It is important to mention that the outcomes of the k-means clustering algorithm are greatly affected by the initial placement of the centroids. This is because the previously mentioned cost function can have numerous local optima, which do not always correspond to real clusters.

We then use the following code to perform qualitative evaluation (plot the Ground Truth and the first Principal Component vs the clustering results):

```
% Reshape the first PC back to the original image dimensions for visualization
Z_total_cube = reshape(PC1_reshaped, p, n);

% Combine the clustering results for both k=6 and k=7 into a matrix
Total_idx_kmeans = [Best_clustering_indices{1}, Best_clustering_indices{2}];

% Plot the Ground Truth, PC1, and clustering results
clust_eval(Total_idx_kmeans, 'k-means', Z_total_cube, salinas_gt, existed_L);

function [] = clust_eval(Total_idx,algorithm,Z_total_cube,Salinas_Labels,existed_L)
    figure;
    colormap("jet");
    subplot(2,2,1);
    imagesc(Salinas_Labels)
    axis off
    title("Ground Truth")
    colorbar
    subplot(2,2,2)
```

```
    imagesc(Z_total_cube(:,:,1))
    axis off
    title("PC1")
    count  = 3;
    for i = 1:(size(Total_idx,2))
        hold on
        cl_label = Total_idx(:,i);
        cl_label_tot=zeros(220*120,1);
        cl_label_tot(existed_L)=cl_label;
        im_cl_label=reshape(cl_label_tot,220,120);
        subplot(2,2,count);
        imagesc(im_cl_label)
        axis off
        title(strcat(algorithm,' | Clusters: ', int2str(length(unique(cl_label)))))
        count  = count + 1;
    end
end
```

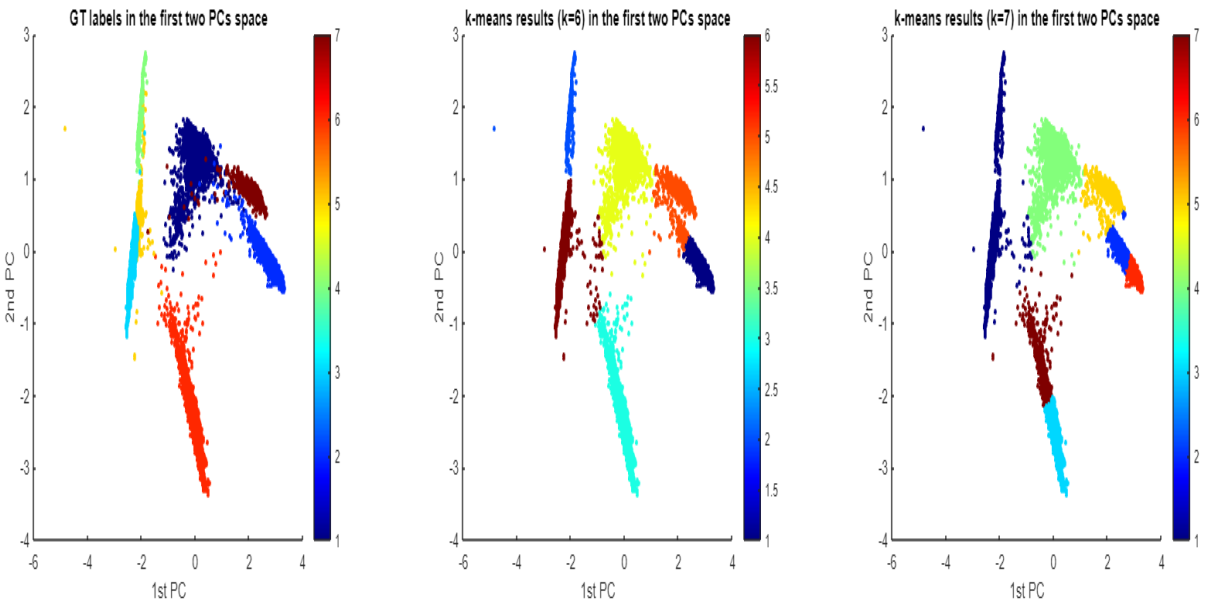


The figure demonstrates that with six clusters, the k-means result shows some mismatch with the ground truth, indicating potential under-segmentation, while seven clusters result in a segmentation that more closely matches the ground truth, although some minor over-segmentation is apparent.

K-means clustering aims to reduce the total squared distances between the points in a cluster and the cluster's center (mean), leading to clusters of roughly equal size. However, this can cause issues with accurately representing the true shape of the clusters. When more than the computed physical number of cluster centers are used, k-means often divides elongated, ellipsoid clusters into multiple sections to minimize its cost function.

To plot the k-means clustering results vs the Ground truth PCA in the space of the first 2 PCs, we use the following code:

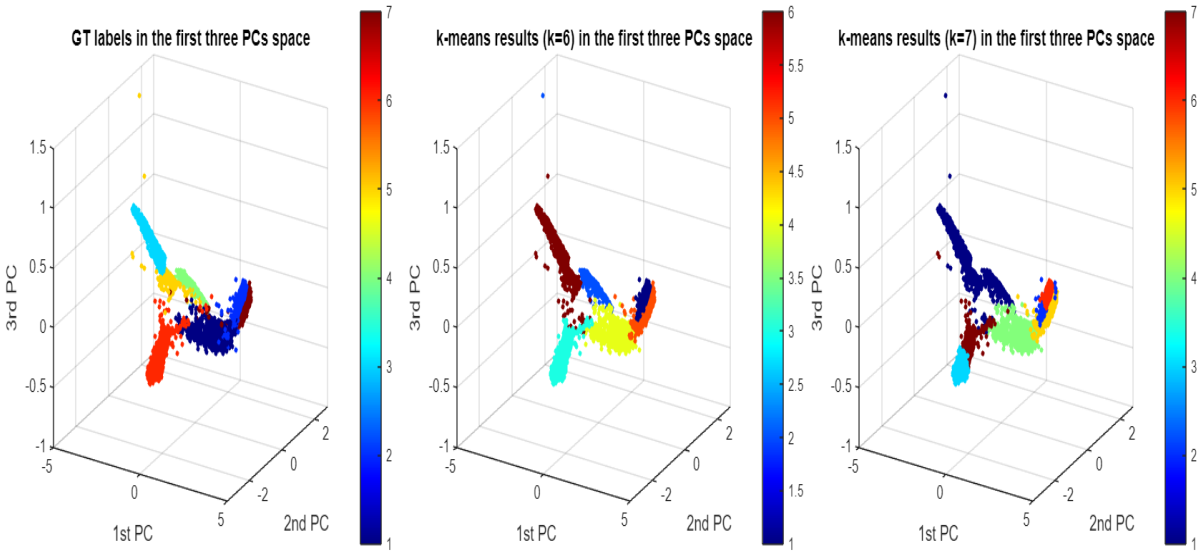
```
% Plot ground truth labels and clustering results for the first two principal components
if DISPLAY_FIGURES
    % Set the figure size
    figure('Position', [100, 100, 1200, 300]); % You can adjust the [left, bottom, width, height] as needed

    % Ground truth labels
    subplot('Position', [0.05, 0.1, 0.28, 0.8]); % [left, bottom, width, height] in normalized units
    scatter(Z(:,1), Z(:,2), 8, Ground_truth, 'filled', 'MarkerEdgeAlpha', 0.9);
    title('GT labels in the first two PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    colormap('jet');
    colorbar;
    axis square;

    % Clustering results for k=6
    subplot('Position', [0.37, 0.1, 0.28, 0.8]);
    scatter(Z(:,1), Z(:,2), 8, Total_idx_kmeans(:,1), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('k-means results (k=6) in the first two PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    colormap('jet');
    colorbar;
    axis square;

    % Clustering results for k=7
    subplot('Position', [0.69, 0.1, 0.28, 0.8]);
    scatter(Z(:,1), Z(:,2), 8, Total_idx_kmeans(:,2), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('k-means results (k=7) in the first two PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    colormap('jet');
    colorbar;
    axis square;
end
```



The comparison of ground truth labels with k-means clustering in the first two principal components reveals that while k-means with 6 clusters captures the general structure of the data, some overlap and under-segmentation are evident. Increasing the cluster count to 7 leads to over-segmentation, where additional, non-existent classes seem to be inferred. These visualizations highlight the sensitivity of k-means to the choice of k and suggest the trial of different approaches to balance the trade-off between under- and over-segmentation.

To plot the k-means clustering results vs the Ground truth PCA in the space of the first 3 PCs, we use the following code:

```
% Plot ground truth labels and clustering results for the first three principal components
if DISPLAY_FIGURES
    % Set the figure size
    figure('Position', [100, 100, 1200, 400]); % You may need to adjust the height to accommodate 3D plots

    % Ground truth labels in 3D
    subplot('Position', [0.05, 0.1, 0.28, 0.8]); % [left, bottom, width, height] in normalized units
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Ground_truth, 'filled', 'MarkerEdgeAlpha', 0.9);
    title('GT labels in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]); % Adjust the view angle for better visibility

    % Clustering results for k=6 in 3D
    subplot('Position', [0.37, 0.1, 0.28, 0.8]);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Total_idx_kmeans(:,1), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('k-means results (k=6) in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]); % Adjust the view angle for better visibility

    % Clustering results for k=7 in 3D
    subplot('Position', [0.69, 0.1, 0.28, 0.8]);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Total_idx_kmeans(:,2), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('k-means results (k=7) in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]); % Adjust the view angle for better visibility
end
```



In the visualizations of ground truth and k-means clustering in three principal components space, the complexity of data structure is more pronounced. The ground truth plot indicates that some classes share closer spectral characteristics, evident when introducing the third PC. K-means clustering with k=6 and k=7 demonstrates increasing complexity and potential overfitting, respectively, as clusters that were distinct in two PCs become more so in three. The third PC reveals overlaps and subdivisions within clusters, suggesting over-segmentation at k=7 and possibly the presence of subgroups.

4.1.2 Fuzzy C-Means

In contrast to k-means, fuzzy c-means (FCM) utilizes a fuzzy approach with a fuzziness factor, denoted as q. This allows each data point to be associated with a cluster to a certain extent. The cost function of FCM bears resemblance to that of k-means and is expressed as $L_q(U,\Theta) = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij}^{q} d(x_i,\theta_j) + \sum_{i=1}^{N}\lambda_i(\sum(_{j=1}^{m}u_{ij} - 1)$, where data points have membership values ranging from 0 to 1. Typically, the value chosen for q is 2. The specific algorithm used here follows the procedure outlined above.

We used the following code:

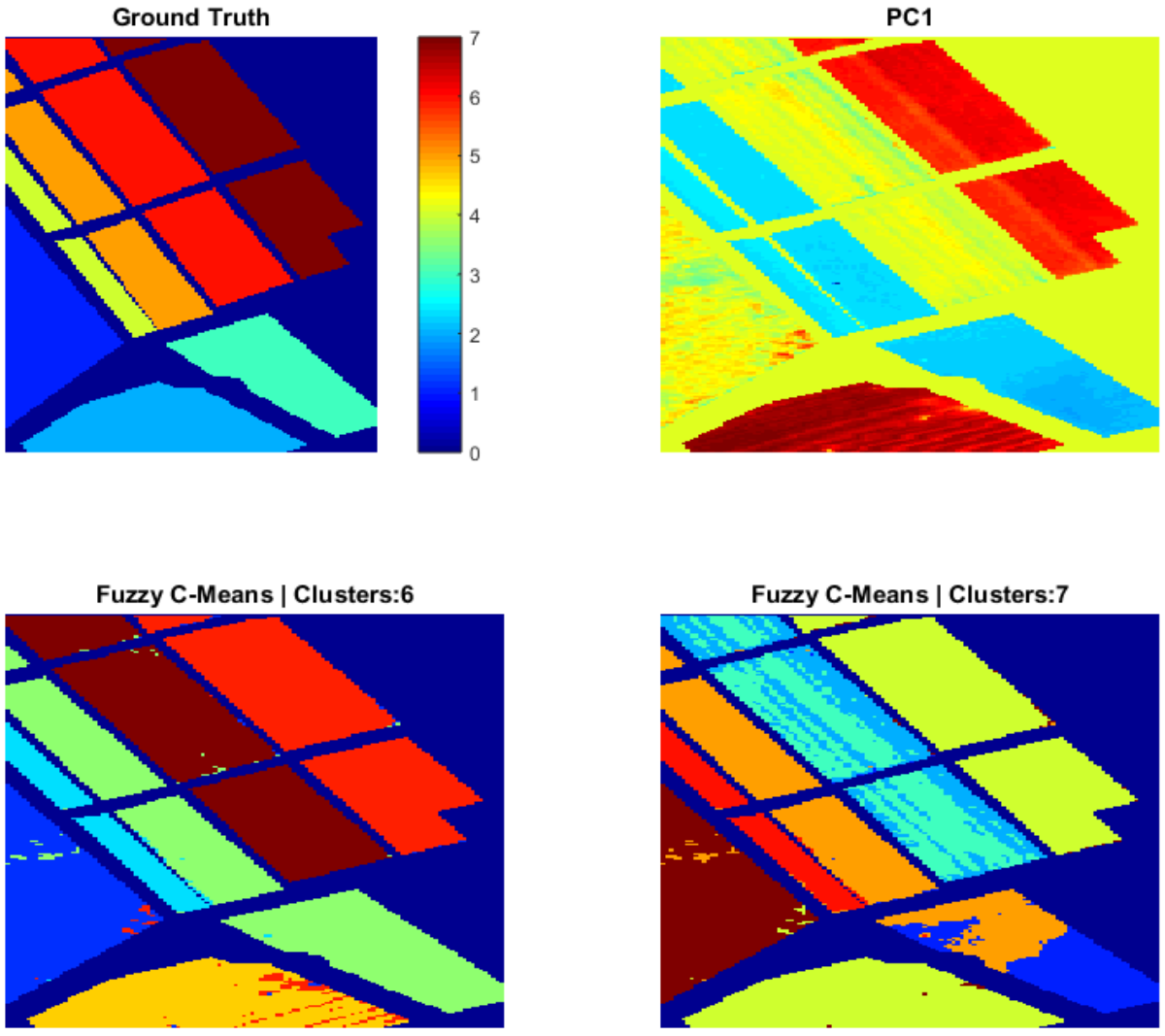
```
% Initialize variables to store the best silhouette scores and corresponding seeds for Fuzzy C-Means
Best_silhouette_score_FCM = zeros(1, 15);
Best_seed_FCM = 0;
numClustersOptions = [6, 7];
Best_clustering_indices_FCM = cell(1, length(numClustersOptions));

% Degree of fuzziness q = 2
options = [2; 100; 1e-5; 0]; % In order: [degree of fuzziness; iterations; error cutoff; verbosity]

% Iterate over seed values for Fuzzy C-Means
for seed = 1:15
    rng(seed, 'twister'); % Set the seed for reproducibility
    temp_clustering_indices_FCM = cell(1, length(numClustersOptions));
    silhouette_scores_FCM = zeros(1, length(numClustersOptions));

    % Perform Fuzzy C-Means clustering and silhouette analysis for each number of clusters
    for clusterOption = 1:length(numClustersOptions)
        currentNumClusters = numClustersOptions(clusterOption);
        [center, U] = fcm(Z, currentNumClusters, options);
        [~, maxU_index] = max(U, [], 1); % Find the index of the max membership value for each data point
        s = silhouette(Z, maxU_index', 'sqeuclidean');
        silhouette_scores_FCM(clusterOption) = mean(s);
        temp_clustering_indices_FCM{clusterOption} = maxU_index'; % Store the clustering indices for this k
    end

    % Compute the average silhouette score for the current seed
    average_silhouette_score_FCM = mean(silhouette_scores_FCM);

    % Store the seed, its score, and clustering indices if it is the best so far for Fuzzy C-Means
    if average_silhouette_score_FCM > max(Best_silhouette_score_FCM)
        Best_silhouette_score_FCM = silhouette_scores_FCM;
        Best_seed_FCM = seed;
        Best_clustering_indices_FCM = temp_clustering_indices_FCM; % Store the best clustering indices for each k
    end
end

% Display the best overall seed and the corresponding silhouette scores for Fuzzy C-Means
disp(['Best overall seed for Fuzzy C-Means: ' num2str(Best_seed_FCM)]);
disp('Silhouette scores for Fuzzy C-Means with the best seed:');
disp(array2table(Best_silhouette_score_FCM, 'VariableNames', {'k6', 'k7'}));
```

The best performance of the FCM algorithm, after iterating through 15 seeds, was obtained from random seed 2, with Silhouette scores: 0.87722 (for the 6 clusters case) and 0.77538 (for the 7 clusters case).

The FCM algorithm excels in situations where data points are not distinctly separable and might be part of several clusters. This method is especially effective when there is some overlap or ambiguity in the data points, making it unsuitable to rigidly assign them to just one cluster. Data points situated near a cluster center show a high degree of belonging to that cluster. Conversely, points that are distant from any cluster center and near the edges of clusters exhibit a much lower sense of belonging.

We then use the following code to perform qualitative evaluation (plot the Ground Truth and the first Principal Component vs the clustering results):

```
% Combine the best clustering results for both k=6 and k=7 into a matrix for Fuzzy C-
    Means
Total_idx_FCM = [Best_clustering_indices_FCM{1}, Best_clustering_indices_FCM{2}];

% Reshape the first PC back to the original image dimensions for visualization
Z_total_cube_FCM = reshape(PC1_reshaped, p, n);

% Plot the Ground Truth, PC1, and FCM clustering results
clust_eval(Total_idx_FCM, 'Fuzzy C-Means', Z_total_cube_FCM, salinas_gt, existed_L);
```

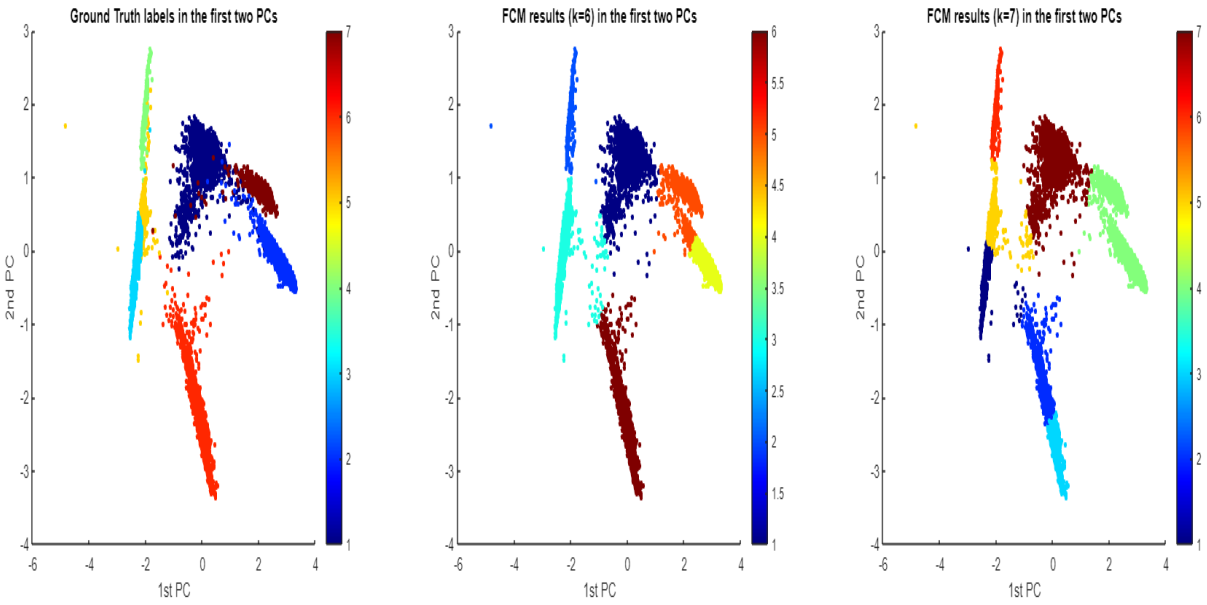


The figure demonstrates that FCM with 6 clusters merges some regions, suggesting under-segmentation, whereas with 7 clusters, it closely aligns with the ground truth but may over-segment. This indicates that the cluster count significantly influences the clustering accuracy in relation to the true labels.

FCM clustering yields similar results to k-means, capturing the first principal component and homogeneous areas effectively for k=7 clusters. This similarity is due to FCM's preference for compact clusters of similar sizes, a trait shared with k-means, which is influenced by the initial placement of cluster centers. Both algorithms aim to minimize the distance between points and their respective cluster centers, leading to the formation of compact clusters, hence the nearly identical clustering results between them.

To plot the FCM clustering results vs the Ground truth PCA in the space of the first 2 PCs, we use the following code:

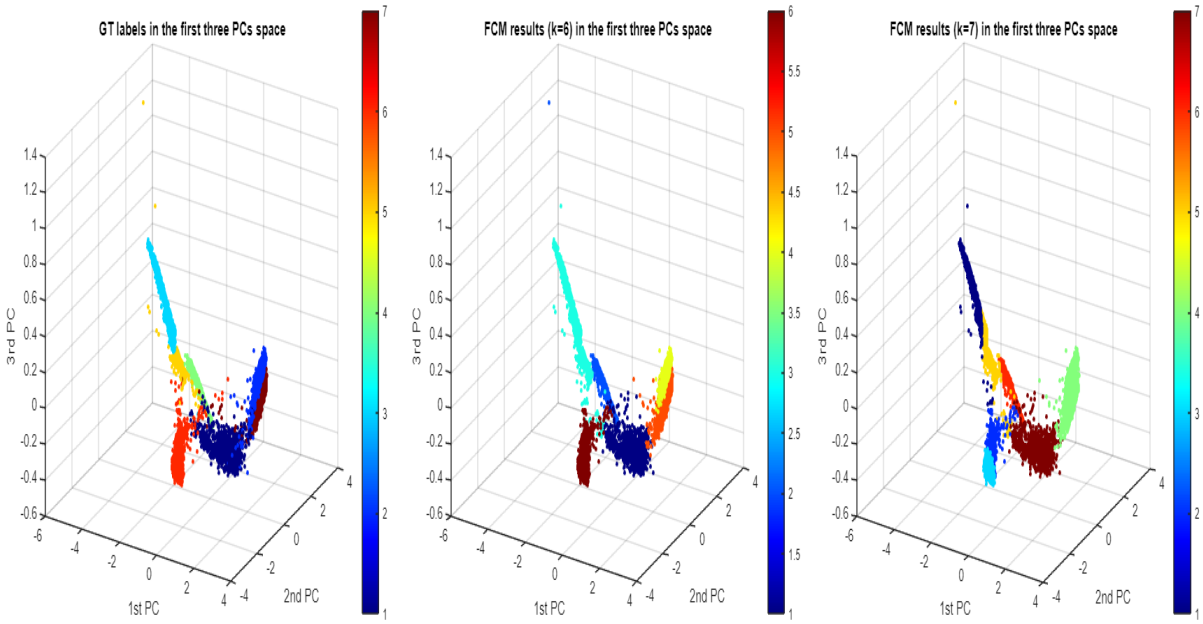
```
% Plot ground truth labels and FCM clustering results for the first two principal
    components
if DISPLAY_FIGURES
    % Set the figure size to accommodate three plots in a row
    figure('Position', [100, 100, 1500, 400]); % Adjust the [left, bottom, width, height
    ] as needed

    % Ground truth labels
    subplot(1, 3, 1, 'Position', [0.05, 0.1, 0.27, 0.8]); % [left, bottom, width, height
    ] in normalized units
    scatter(Z(:,1), Z(:,2), 10, Ground_truth, 'filled', 'MarkerEdgeAlpha', 0.9);
    title('Ground Truth labels in the first two PCs');
    xlabel('1st PC');
    ylabel('2nd PC');
    axis square;
    set(gca, 'Colormap', jet);
    colorbar;

    % FCM clustering results for k=6
    subplot(1, 3, 2, 'Position', [0.37, 0.1, 0.27, 0.8]);
    scatter(Z(:,1), Z(:,2), 10, Total_idx_FCM(:,1), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('FCM results (k=6) in the first two PCs');
    xlabel('1st PC');
    ylabel('2nd PC');
    axis square;
    set(gca, 'Colormap', jet);
    colorbar;

    % FCM clustering results for k=7
    subplot(1, 3, 3, 'Position', [0.69, 0.1, 0.27, 0.8]);
    scatter(Z(:,1), Z(:,2), 10, Total_idx_FCM(:,2), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('FCM results (k=7) in the first two PCs');
    xlabel('1st PC');
    ylabel('2nd PC');
    axis square;
    set(gca, 'Colormap', jet);
    colorbar;
end
```



The center plot with k=6 clusters shows FCM's attempt at grouping the data; the clusters appear somewhat mixed, suggesting slight under-segmentation. In contrast, the rightmost plot with k=7 clusters displays a tendency towards over-segmentation, as the algorithm assigns multiple clusters to single ground truth classes.

To plot the FCM clustering results vs the Ground truth PCA in the space of the first 3 PCs, we use the following code:

```
% Plot ground truth labels and FCM clustering results for the first three principal components
if DISPLAY_FIGURES
    % Set the figure size
    figure('Position', [100, 100, 1200, 400]); % You may need to adjust the height to accommodate 3D plots

    % Ground truth labels in 3D
    subplot('Position', [0.05, 0.1, 0.28, 0.8]); % [left, bottom, width, height] in normalized units
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Ground_truth, 'filled');
    title('GT labels in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]); % Adjust the view angle for better visibility

    % FCM clustering results for k=6 in 3D
    subplot('Position', [0.37, 0.1, 0.28, 0.8]);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Total_idx_FCM(:,1), 'filled');
    title('FCM results (k=6) in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]); % Adjust the view angle for better visibility

    % FCM clustering results for k=7 in 3D
    subplot('Position', [0.69, 0.1, 0.28, 0.8]);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Total_idx_FCM(:,2), 'filled');
    title('FCM results (k=7) in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]); % Adjust the view angle for better visibility
end
```



FCM clustering with k=6 clusters presents a reasonable separation of clusters, but there's some ambiguity and potential overlap, especially in regions where the ground truth classes are closely situated. With k=7 clusters, the FCM algorithm seems to over-partition the data, suggesting an over-segmentation similar to what was observed with k-means, which might indicate the algorithm is detecting more clusters than actually exist. The addition of the third PC helps to differentiate between some classes but also highlights the challenges in defining clear boundaries between them, emphasizing the nuanced nature of spectral data clustering.

4.1.3 Possibilistic C-Means

The Possibilistic C-Means (PCM) algorithm, like FCM, employs a fuzzy approach, but differs in that each point's membership degree to a cluster is independent from its membership to all other clusters. Therefore, its cost function, akin to FCM's, includes a regularization term defined by the user, denoted as η, which is linked to the clusters' variance. This helps avoid solutions that are constantly zero. The cost function is given by:

$$L_q(U,\Theta) = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij}^q d(x_i,\theta_j) + \sum_{j=1}^{N} \eta_j (\sum(_{i=1}^{m} 1 - u_{ij})^q$$

Regarding η, it can be determined in two ways:

1. As an outcome of running the FCM algorithm, where $\eta_j = \frac{\sum_{i=1}^{N} u_{ij}^q d(x_i,\theta_j)}{\sum_{i=1}^{N} u_{ij}}^q$
2. Alternatively, $\eta = \frac{1}{N}\sum_{i=1}^{N} ||x_i - \bar{x}||^2$

To ensure that PCM's results are not influenced by those of FCM, we calculate η using the second method. The initialization method chosen involves selecting vectors X that are maximally distant from each other. Here, the specific implementation of the PCM algorithm was taken from (c) 2010 S. Theodoridis, A. Pikrakis, K. Koutroumbas, D. Cavouras.

We used the following code:

```
% Initialize variables for PCM
Best_silhouette_score_PCM = zeros(1, 15);
Best_seed_PCM = 0;
numClustersOptions = [6, 7];
Best_clustering_indices_PCM = cell(1, length(numClustersOptions));

% Transpose Z for compatibility with possibi function
Z_transposed = Z';

% Iterate over seed values for Possibilistic C-Means
for seed = 1:15
    rng(seed, 'twister'); % Set the seed for reproducibility
    temp_clustering_indices_PCM = cell(1, length(numClustersOptions));
    silhouette_scores_PCM = zeros(1, length(numClustersOptions));

    % Calculate eta for PCM
    mean_Z = mean(Z, 2);
    diff_Z = bsxfun(@minus, Z, mean_Z);
    eta = mean(sum(diff_Z .^ 2, 2));

    % Perform Possibilistic C-Means clustering and silhouette analysis for each number
    of clusters
    for clusterOption = 1:length(numClustersOptions)
        currentNumClusters = numClustersOptions(clusterOption);
        eta_m = ones(1, currentNumClusters) * eta;

        % Perform PCM clustering using the 'possibi' function with transposed Z
        [U, ~] = possibi(Z_transposed, currentNumClusters, eta_m, 2, seed, 3, 0.001); %
    Adjust the 'possibi' function parameters as needed
        [~, maxU_index] = max(U, [], 2);

        % Calculate the silhouette score
        s = silhouette(Z, maxU_index, 'sqeuclidean');
        silhouette_scores_PCM(clusterOption) = mean(s);
```

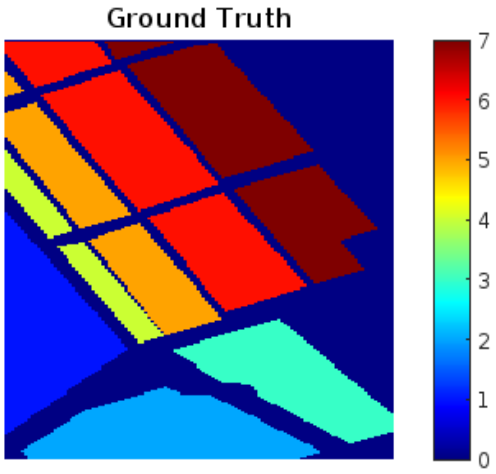
```
        temp_clustering_indices_PCM{clusterOption} = maxU_index;
    end

    % Compute the average silhouette score for the current seed
    average_silhouette_score_PCM = mean(silhouette_scores_PCM);

    % Store the seed, its score, and clustering indices if it is the best so far for
    Possibilistic C-Means
    if average_silhouette_score_PCM > max(Best_silhouette_score_PCM)
        Best_silhouette_score_PCM = silhouette_scores_PCM;
        Best_seed_PCM = seed;
        Best_clustering_indices_PCM = temp_clustering_indices_PCM; % Store the best
    clustering indices for each k
    end
end

% Display the best overall seed and the corresponding silhouette scores for
    Possibilistic C-Means
disp(['Best overall seed for Possibilistic C-Means: ' num2str(Best_seed_PCM)]);
disp('Silhouette scores for Possibilistic C-Means with the best seed:');
disp(array2table(Best_silhouette_score_PCM, 'VariableNames', {'k6', 'k7'}));
```

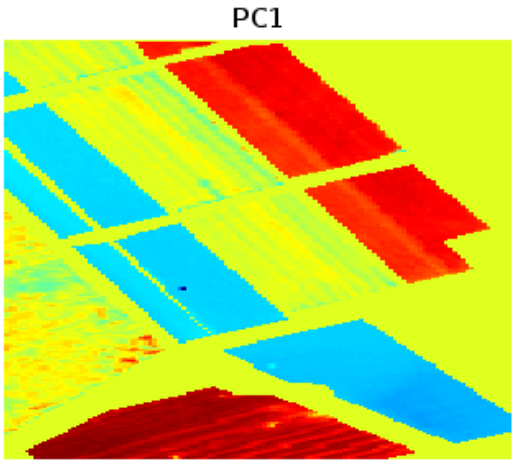
The best performance of the PCM algorithm, after iterating through 15 seeds, was obtained from random seed 1, with Silhouette scores: 0.29306 (for the 6 clusters case) and 0.29296 (for the 7 clusters case).

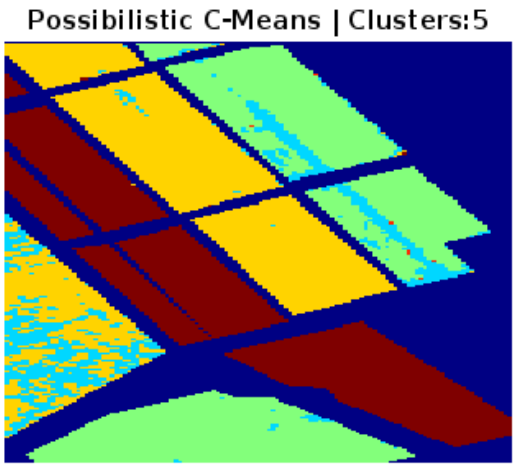
PCM assigns higher degrees of membership to points in densely populated regions, akin to FCM, while those in sparser, noisier areas receive lower degrees. A distinctive feature of PCM, however, is its ability to not assign some points to any cluster at all, which can enhance the clustering quality amidst noisy data. Additionally, PCM is flexible in adjusting the count of cluster centers, which can lead to improved partitioning. However, if the chosen number of clusters is lower than the actual count in the data, PCM may fail to position cluster centers in all dense regions. Conversely, if the number is set too high, it may fragment natural clusters into smaller subsets.

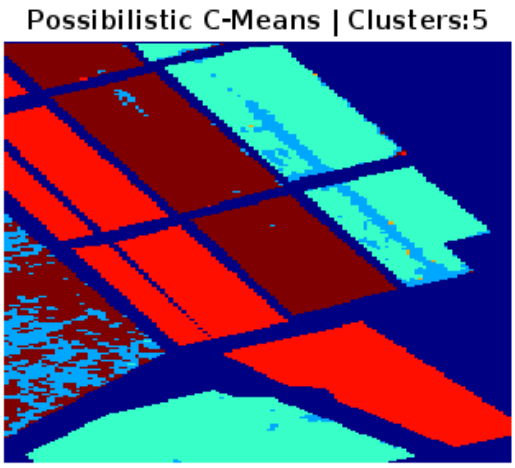
We then use the following code to perform qualitative evaluation (plot the Ground Truth and the first Principal Component vs the clustering results):

```
% Combine the best clustering results for both k=6 and k=7 into a matrix for
    Possibilistic C-Means
Total_idx_PCM = [Best_clustering_indices_PCM{1}, Best_clustering_indices_PCM{2}];

% Reshape the first PC back to the original image dimensions for visualization
Z_total_cube_PCM = reshape(PC1_reshaped, p, n);

% Plot the Ground Truth, PC1, and PCM clustering results
clust_eval(Total_idx_PCM, 'Possibilistic C-Means', Z_total_cube_PCM, salinas_gt,
    existed_L);
```









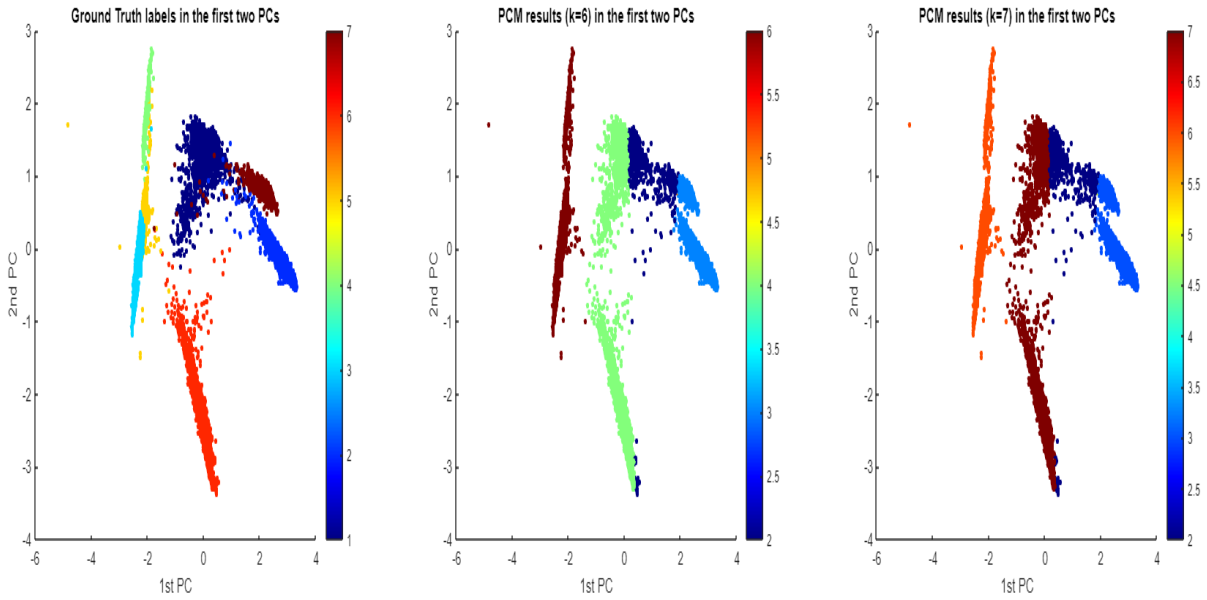
Both PCM attempts were intended to segment into 6 and 7 clusters but resulted in only 5 clusters each. This discrepancy suggests an anomaly in the clustering process or a possible coding error that restricted the output to 5 clusters regardless of the intended number. The clustering outcomes seem to oversimplify the complexity of the data, leading to an inaccurate representation of the ground truth.

PCM also exhibits the same random representative initialization sensitivity as k-means and fuzzy c-means, as described previously.

To plot the PCM clustering results vs the Ground truth PCA in the space of the first 2 PCs, we use the following code:

```
if DISPLAY_FIGURES
    % Set the figure size to accommodate three plots in a row
    figure('Position', [100, 100, 1500, 400]); % Adjust as needed

    % Ground truth labels
    subplot(1, 3, 1, 'Position', [0.05, 0.1, 0.27, 0.8]);
    scatter(Z(:,1), Z(:,2), 10, Ground_truth, 'filled', 'MarkerEdgeAlpha', 0.9);
    title('Ground Truth labels in the first two PCs');
    xlabel('1st PC');
    ylabel('2nd PC');
    axis square;
    colormap('jet');
    colorbar;

    % PCM clustering results for k=6
    subplot(1, 3, 2, 'Position', [0.37, 0.1, 0.27, 0.8]);
    scatter(Z(:,1), Z(:,2), 10, Total_idx_PCM(:,1), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('PCM results (k=6) in the first two PCs');
    xlabel('1st PC');
```

```
    ylabel('2nd PC');
    axis square;
    colormap('jet');
    colorbar;

    % PCM clustering results for k=7
    subplot(1, 3, 3, 'Position', [0.69, 0.1, 0.27, 0.8]);
    scatter(Z(:,1), Z(:,2), 10, Total_idx_PCM(:,2), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('PCM results (k=7) in the first two PCs');
    xlabel('1st PC');
    ylabel('2nd PC');
    axis square;
    colormap('jet');
    colorbar;
end
```

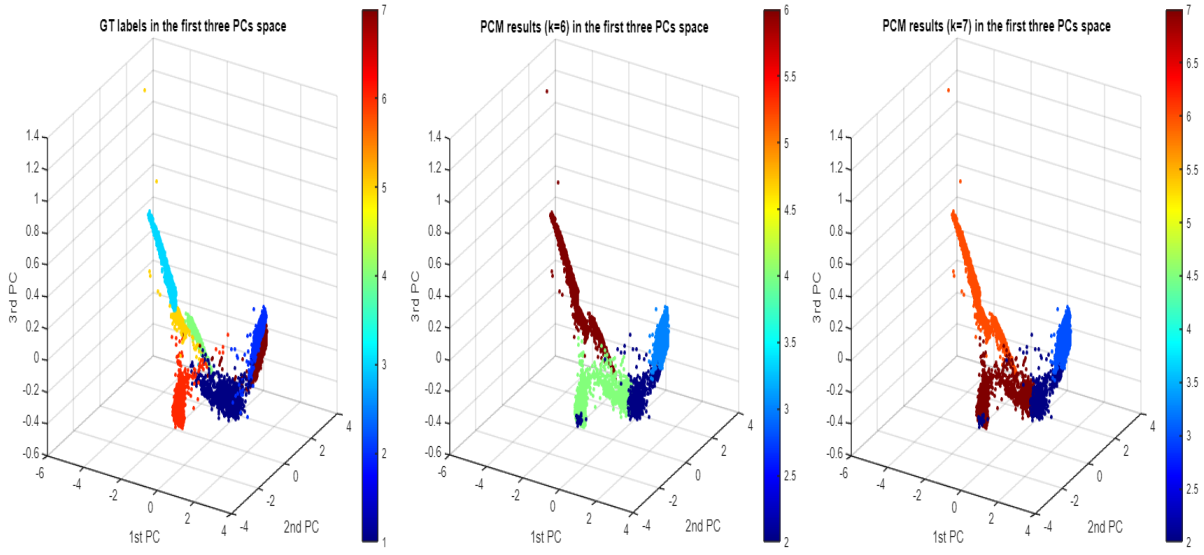


The center and rightmost plots illustrate the outcomes of PCM clustering with 6 and 7 clusters, respectively. However, it's important to note that the exact number of clusters in the algorithm's results does not match the intended numbers (6 and 7), indicating an anomaly in the clustering process that needs further investigation.

To plot the PCM clustering results vs the Ground truth PCA in the space of the first 3 PCs, we use the following code:

```
if DISPLAY_FIGURES
    % Set the figure size for 3D plots
    figure('Position', [100, 100, 1200, 400]); % Adjust as needed

    % Ground truth labels in 3D
    subplot('Position', [0.05, 0.1, 0.28, 0.8]);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Ground_truth, 'filled');
    title('GT labels in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]);

    % PCM clustering results for k=6 in 3D
    subplot('Position', [0.37, 0.1, 0.28, 0.8]);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Total_idx_PCM(:,1), 'filled');
    title('PCM results (k=6) in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
```

```
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]);

    % PCM clustering results for k=7 in 3D
    subplot('Position', [0.69, 0.1, 0.28, 0.8]);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Total_idx_PCM(:,2), 'filled');
    title('PCM results (k=7) in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]);
end
```



The middle and rightmost plots show the PCM clustering outcomes for 6 and 7 clusters respectively. However, it's important to note that the exact number of clusters in the algorithm's results does not match the intended numbers (6 and 7), indicating an anomaly in the clustering process that needs further investigation.

4.1.4 Probabilistic C-Means

Probabilistic C-Means (PBCM) operates on the principle that the dataset is composed of a limited number of Gaussian distributions with unknown parameters, corresponding to distinct clusters. The exact cluster to which each point belongs is uncertain, leading to an estimation problem for the parameters of each Gaussian distribution, factoring in the unobserved variables. To resolve this and converge to a local optimum, the Expectation Maximization algorithm is employed, which in this implementation is executed using MATLAB's *fitgmdist* function. It is noteworthy that the silhouette score, commonly used in clustering validation, is not as reliable for probabilistic models as the negative log-likelihood, which more effectively evaluates the fit of a probabilistic model to a dataset. This approach does not presume independence among features, hence the covariance matrix Σ is not restricted to $\sigma^2 I$.

We used the following code:

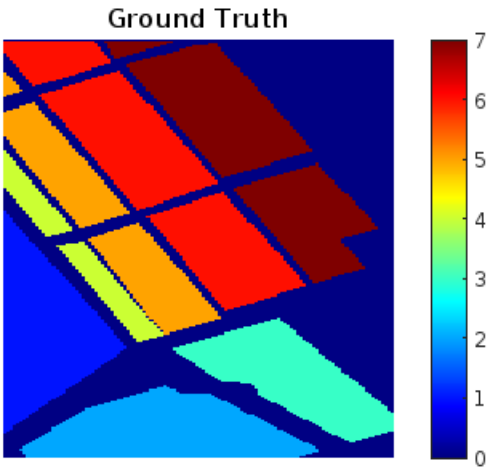
```
% Initialize variables for PBCM
Best_negLogLikelihood_PBCM = inf(1, 15);  % Store the best negative log-likelihood for
    each seed
Best_seeds_PBCM = zeros(1, 15);  % Store the best seeds
numClustersOptions = [6, 7];  % Cluster options
Best_clustering_indices_PBCM = cell(length(numClustersOptions), 15);  % Store clustering
     results for each k and each seed

% Iterate over seeds and cluster options
```

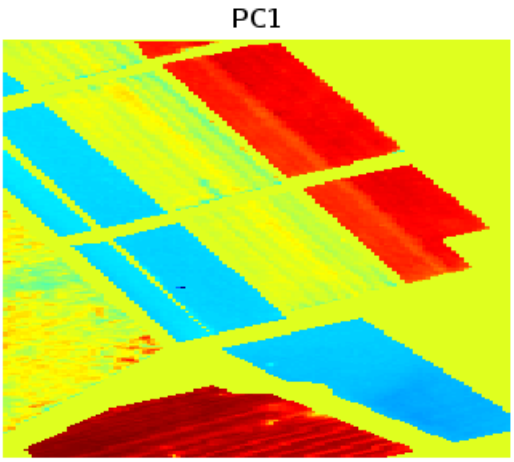
```
for seed = 1:15
    rng(seed, 'twister'); % Set the seed for reproducibility

    for clusterOption = 1:length(numClustersOptions)
        currentNumClusters = numClustersOptions(clusterOption);
        GMModel = fitgmdist(Z, currentNumClusters, 'CovarianceType', 'full', 'Replicates', 1);

        % Cluster assignment and model evaluation
        [~, nLogL] = cluster(GMModel, Z);

        % Update the best model based on negative log-likelihood
        if nLogL < Best_negLogLikelihood_PBCM(seed)
            Best_negLogLikelihood_PBCM(seed) = nLogL;
            Best_seeds_PBCM(seed) = seed;
            Best_clustering_indices_PBCM{clusterOption, seed} = cluster(GMModel, Z);
        end
    end
end

% Find the overall best seed based on the negative log-likelihood
[~, bestSeedIndex] = min(Best_negLogLikelihood_PBCM);
bestSeed = Best_seeds_PBCM(bestSeedIndex);

% Display the best overall seed and the corresponding negative log-likelihoods
disp(['Best overall seed: ' num2str(bestSeed)]);
for clusterOption = 1:length(numClustersOptions)
    disp(['Negative Log Likelihood for ' num2str(numClustersOptions(clusterOption)) ' clusters: ' num2str(Best_negLogLikelihood_PBCM(bestSeedIndex))]);
end
```

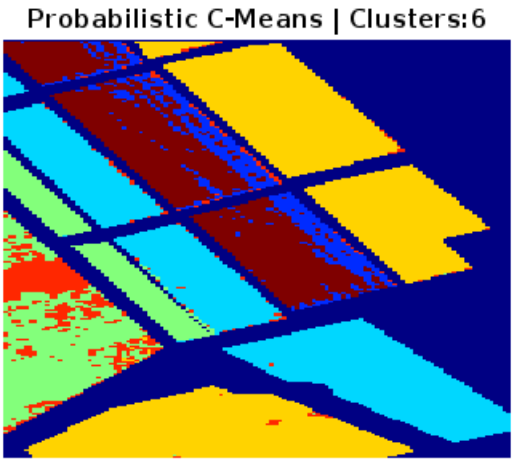
The best performance of the PBCM algorithm, after iterating through 15 seeds, was obtained from random seed 14, with Negative Log Likelihoods (here we look for the lowest value): -25234.6713 (for the 6 clusters case) and -25234.6713 (for the 7 clusters case).

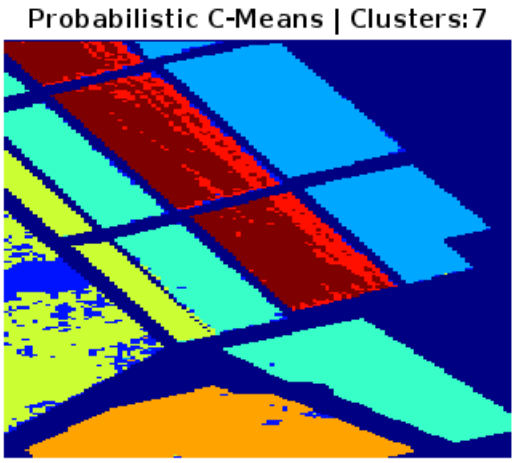
We then use the following code to perform qualitative evaluation (plot the Ground Truth and the first Principal Component vs the clustering results):

```
% Combine the best clustering results for both k=6 and k=7 into a matrix for PBCM
Total_idx_PBCM = [Best_clustering_indices_PBCM{1}, Best_clustering_indices_PBCM{2}];

% Reshape the first principal component back to the original image dimensions for visualization
Z_total_cube_PBCM = reshape(PC1_reshaped, p, n);

% Plot the Ground Truth, first principal component, and PBCM clustering results
clust_eval(Total_idx_PBCM, 'Probabilistic C-Means', Z_total_cube_PBCM, salinas_gt, existed_L);
```









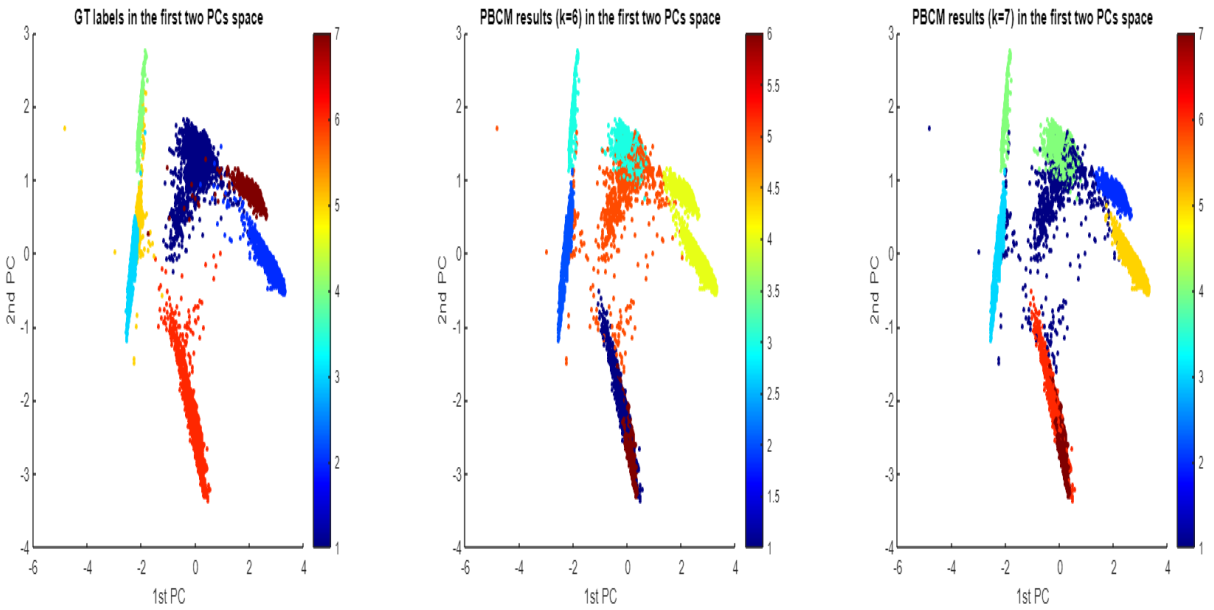
The PBCM results for 6 clusters seem to under-segment the data, failing to distinguish between some of the classes, while the results for 7 clusters offer a closer match to the ground truth but may suffer from slight over-segmentation in certain areas. The visual representation indicates that PBCM, with its probabilistic approach, is flexible enough to model the complex spatial structures present in hyperspectral images and that choosing the correct number of clusters is crucial for aligning with the true labels.

PBCM with a full covariance matrix often inaccurately predicts ground truth labels and fails to differentiate regions with similar data distributions in a 3-D dataset space. The capability of full covariance PBCM to model complex, non-compact cluster shapes like ellipsoids can potentially enhance clustering performance. However, this complexity brings a higher risk of converging to suboptimal solutions, leading to inconsistent results. Furthermore, a low negative log-likelihood, a common model fit metric, doesn't always correspond to an accurate representation of underlying data structures, as demonstrated by frequent mispredictions. This issue is compounded by the algorithm's tendency to assign similarly shaped data clusters to a single cluster, despite their separation in space.

To plot the PBCM clustering results vs the Ground truth PCA in the space of the first 2 PCs, we use the following code:

```
% Plot ground truth labels and PBCM clustering results for the first two principal
    components
if DISPLAY_FIGURES
    % Set the figure size
    figure('Position', [100, 100, 1200, 300]);

    % Ground truth labels
    subplot('Position', [0.05, 0.1, 0.28, 0.8]);
    scatter(Z(:,1), Z(:,2), 8, Ground_truth, 'filled', 'MarkerEdgeAlpha', 0.9);
    title('GT labels in the first two PCs space');
```

```
    xlabel('1st PC');
    ylabel('2nd PC');
    colormap('jet');
    colorbar;
    axis square;

    % PBCM clustering results for k=6
    subplot('Position', [0.37, 0.1, 0.28, 0.8]);
    scatter(Z(:,1), Z(:,2), 8, Total_idx_PBCM(:,1), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('PBCM results (k=6) in the first two PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    colormap('jet');
    colorbar;
    axis square;

    % PBCM clustering results for k=7
    subplot('Position', [0.69, 0.1, 0.28, 0.8]);
    scatter(Z(:,1), Z(:,2), 8, Total_idx_PBCM(:,2), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('PBCM results (k=7) in the first two PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    colormap('jet');
    colorbar;
    axis square;
end
```

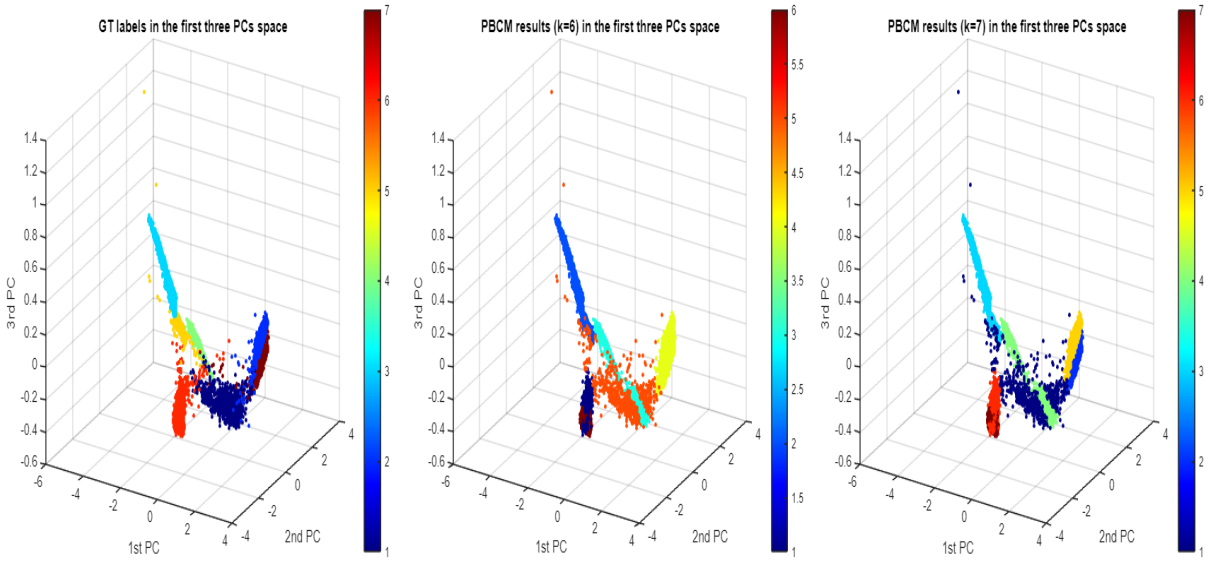


In the middle plot, where k=6, the PBCM algorithm appears to segment the data into distinct clusters, but with some evident mismatches compared to the ground truth labels shown in the left plot. On the right plot, increasing the number of clusters to k=7 leads to finer clustering, which seems to align more closely with certain areas of the ground truth. However, it also suggests potential over-segmentation, as seen by the emergence of additional clusters.

To plot the PBCM clustering results vs the Ground truth PCA in the space of the first 3 PCs, we use the following code:

```
% Plot ground truth labels and PBCM clustering results for the first three principal
    components
if DISPLAY_FIGURES
    % Set the figure size for 3D plots
    figure('Position', [100, 100, 1200, 400]); % Adjust as needed

    % Ground truth labels in 3D
    subplot('Position', [0.05, 0.1, 0.28, 0.8]);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Ground_truth, 'filled', 'MarkerEdgeAlpha', 0.9);
    title('GT labels in the first three PCs space');
    xlabel('1st PC');
```

```
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]); % Adjust the view angle

    % PBCM clustering results for k=6 in 3D
    subplot('Position', [0.37, 0.1, 0.28, 0.8]);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Total_idx_PBCM(:,1), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('PBCM results (k=6) in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]); % Adjust the view angle

    % PBCM clustering results for k=7 in 3D
    subplot('Position', [0.69, 0.1, 0.28, 0.8]);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Total_idx_PBCM(:,2), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('PBCM results (k=7) in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]); % Adjust the view angle
end
```



The center and right plots depict the PBCM clustering outcomes for k=6 and k=7, respectively. Both scenarios exhibit clusters that closely match portions of the ground truth, although some regions show signs of misclassification. The plots indicate that while the PBCM is capable of discerning intricate groupings in multidimensional space, the selection of the number of clusters remains critical for achieving high fidelity to the actual class distributions, with k=7 providing a slightly better resolution than k=6 in this specific case.

4.2 Hierarchical Clustering Algorithms

Hierarchical algorithms, in contrast to the cost function optimization-based clustering methods, yield consistent and deterministic outcomes. This predictability stems from the linkage function which operates on a defined set of rules determined by the chosen method and distance metric, avoiding any stochastic elements. Thus, given the same distance and method, the hierarchical algorithm's output remains constant.

These algorithms employ an agglomerative approach to clustering, treating each data point as its own cluster initially and then progressively merging the nearest clusters in subsequent steps according to their specific methodology and distance metric. However, this process can be inflexible; once clusters are combined in a manner that may not align with the intended goal, there is no way within the same algorithmic framework and metric to rectify such merges.

In the context of clustering within a 3-dimensional space, the Euclidean distance remains a robust choice for measuring proximity. Moreover, algorithms like Ward's method are particularly tailored to capitalize on the Euclidean metric, optimizing their clustering efficacy in such spaces.

4.2.1 Complete-Link Algorithm

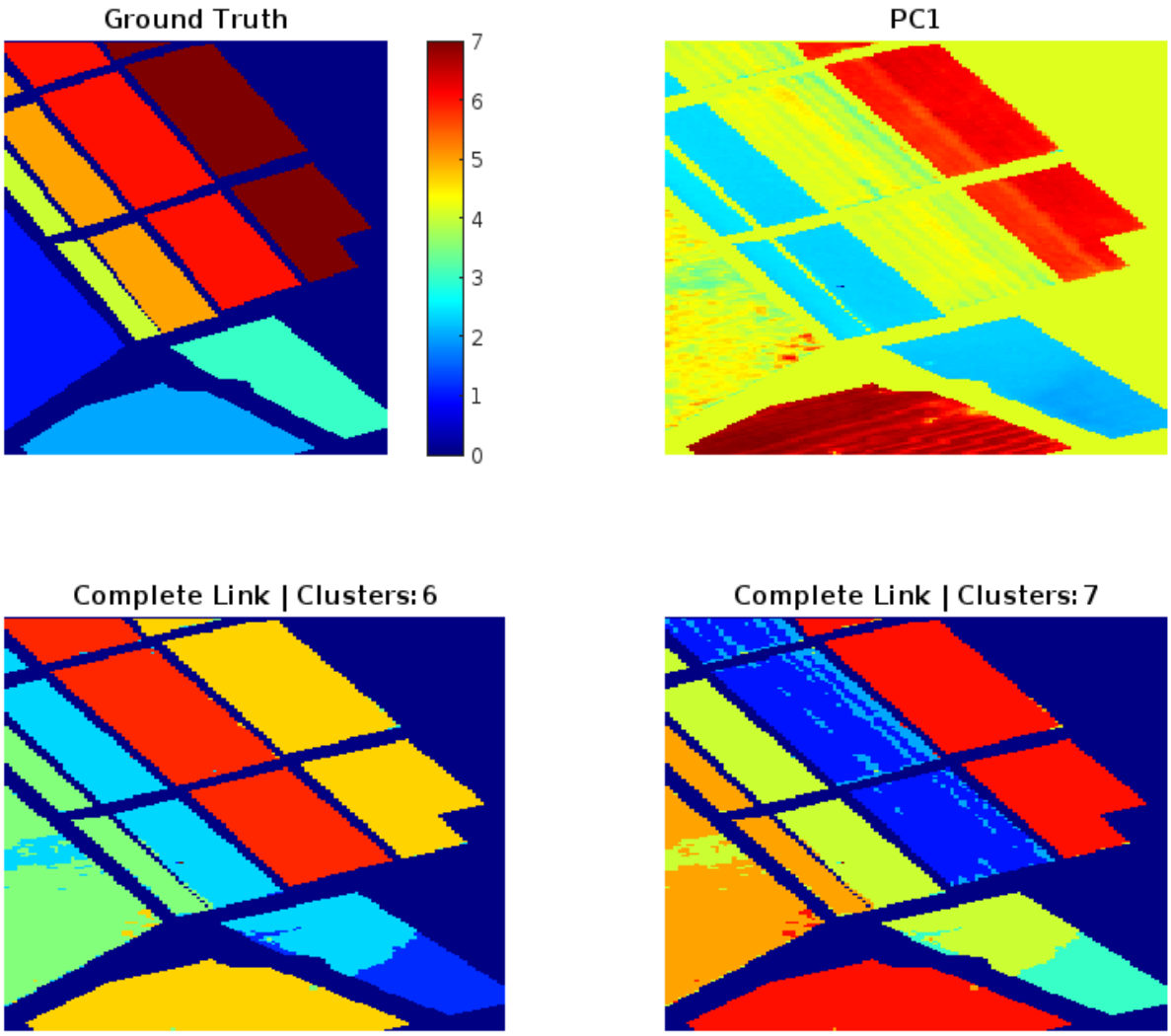
The Complete Link clustering technique uses a strict measure for combining clusters, taking into account the greatest distance between the most distant members of each pair of clusters under consideration at every step of the merging process.

We used the following code:

```
% Perform Complete Link Hierarchical Clustering for specified cluster numbers
Total_idx_CL = [];
for i = [6, 7]
    B = linkage(Z, 'complete', 'euclidean');
    idx = cluster(B, 'maxclust', i);
    Total_idx_CL = cat(2, Total_idx_CL, idx);
end
```

We then use the following code to perform qualitative evaluation (plot the Ground Truth and the first Principal Component vs the clustering results):

```
% Reshape the first PC back to the original image dimensions for visualization
Z_total_cube_CL = reshape(PC1_reshaped, p, n);

% Plot the Ground Truth, PC1, and Complete Link clustering results
clust_eval(Total_idx_CL, 'Complete Link', Z_total_cube_CL, salinas_gt, existed_L);
```

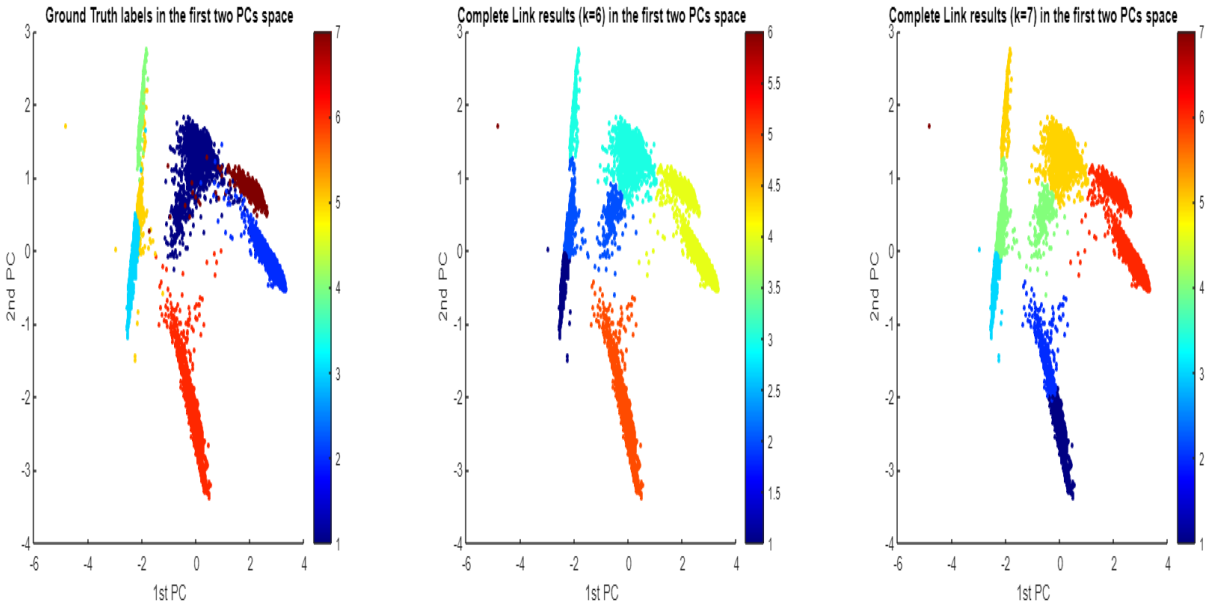


For both k=6 and k=7, the Complete Link algorithm shows a mostly clear delineation of clusters, capturing the variation within the data similarly to the PCA results. The method preserves distinct boundaries between clusters, reflecting a more rigid structure, which is characteristic of the Complete Link approach. However, this rigidity may also lead to less flexibility in accommodating the natural distribution of the data, potentially causing a mismatch with the ground truth labels, especially noticeable in the division of clusters that are contiguous in the ground truth. This highlights the algorithm's tendency towards creating equally sized clusters and its difficulty in handling noise and outliers.

To plot the CL clustering results vs the Ground truth PCA in the space of the first 2 PCs, we use the following code:

```
if DISPLAY_FIGURES
    % Set the figure size for 2D plots
    figure('Position', [100, 100, 1200, 300]);

    % Ground truth labels for the first two principal components
    subplot('Position', [0.05, 0.1, 0.28, 0.8]);
    scatter(Z(:,1), Z(:,2), 8, Ground_truth, 'filled', 'MarkerEdgeAlpha', 0.9);
    title('Ground Truth labels in the first two PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    colormap('jet');
    colorbar;
    axis square;

    % Complete Link clustering results for k=6 in the first two principal components
    subplot('Position', [0.37, 0.1, 0.28, 0.8]);
    scatter(Z(:,1), Z(:,2), 8, Total_idx_CL(:,1), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('Complete Link results (k=6) in the first two PCs space');
```

```
    xlabel('1st PC');
    ylabel('2nd PC');
    colormap('jet');
    colorbar;
    axis square;

    % Complete Link clustering results for k=7 in the first two principal components
    subplot('Position', [0.69, 0.1, 0.28, 0.8]);
    scatter(Z(:,1), Z(:,2), 8, Total_idx_CL(:,2), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('Complete Link results (k=7) in the first two PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    colormap('jet');
    colorbar;
    axis square;
end
```

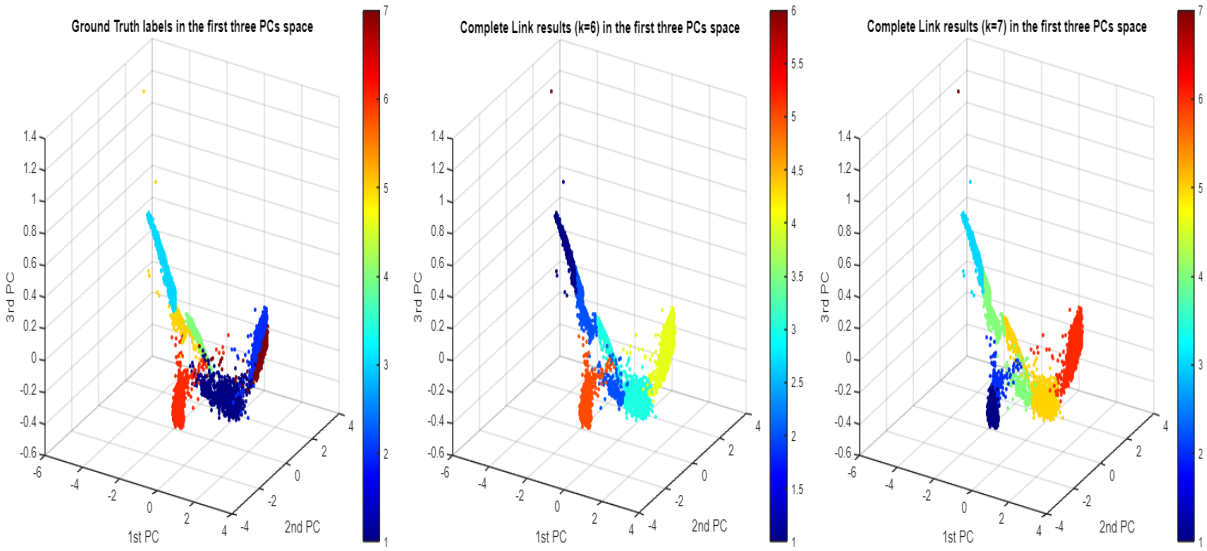


At k=6, the algorithm distinguishes the clusters but with less precision than the ground truth, as seen by the dispersed points and lack of clear boundaries. When k is increased to 7, the algorithm attempts to refine the segmentation but still falls short, with significant misclassification evident. This suggests that while Complete Link is sensitive to the choice of k, neither 6 nor 7 clusters capture the true data structure as effectively as the original labels, indicating that the method might benefit from a different clustering number or approach to better match the ground truth.

To plot the CL clustering results vs the Ground truth PCA in the space of the first 3 PCs, we use the following code:

```
if DISPLAY_FIGURES
    % Set the figure size for 3D plots
    figure('Position', [100, 100, 1200, 400]);

    % Ground truth labels for the first three principal components
    subplot('Position', [0.05, 0.1, 0.28, 0.8]);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Ground_truth, 'filled');
    title('Ground Truth labels in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]);

    % Complete Link clustering results for k=6 in the first three principal components
    subplot('Position', [0.37, 0.1, 0.28, 0.8]);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Total_idx_CL(:,1), 'filled');
    title('Complete Link results (k=6) in the first three PCs space');
```

```
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]);

    % Complete Link clustering results for k=7 in the first three principal components
    subplot('Position', [0.69, 0.1, 0.28, 0.8]);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Total_idx_CL(:,2), 'filled');
    title('Complete Link results (k=7) in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]);
end
```



It is evident that while the algorithm with 6 clusters broadly captures the structure of the data, there are noticeable differences from the ground truth, particularly in the coarser clustering of some regions. With 7 clusters, the algorithm appears to delineate the data with increased granularity, which may align more closely with the ground truth in some areas but also suggests potential over-segmentation in others. This contrast between the two clustering results underscores the impact of the chosen number of clusters on the algorithm's ability to accurately reflect the underlying data distribution.

4.2.2 Weighted Pair Group Method Centroid (WPGMC)

The Weighted Pair Group Method with Centroid Averaging (WPGMC) algorithm is more flexible compared to the Complete Link algorithm. It calculates the distance for the linkage method using an equation that balances the distances between clusters and their sizes. This equation considers the weighted distance from a cluster to a newly formed cluster by taking into account the size of each cluster, and it slightly adjusts the distance based on the proximity of the two clusters being merged:

$$d_{qs} = \frac{n_i}{n_i + n_j} d(C_i, C_s) + \frac{n_j}{n_i + n_j} d(C_j, C_s) - \frac{1}{4} d(C_i, C_j)$$

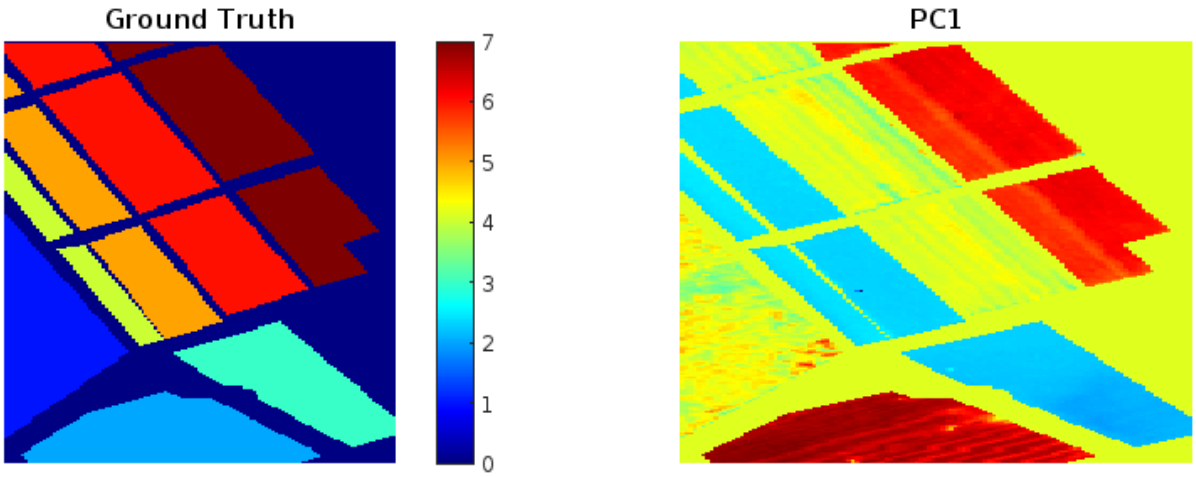
We used the following code:

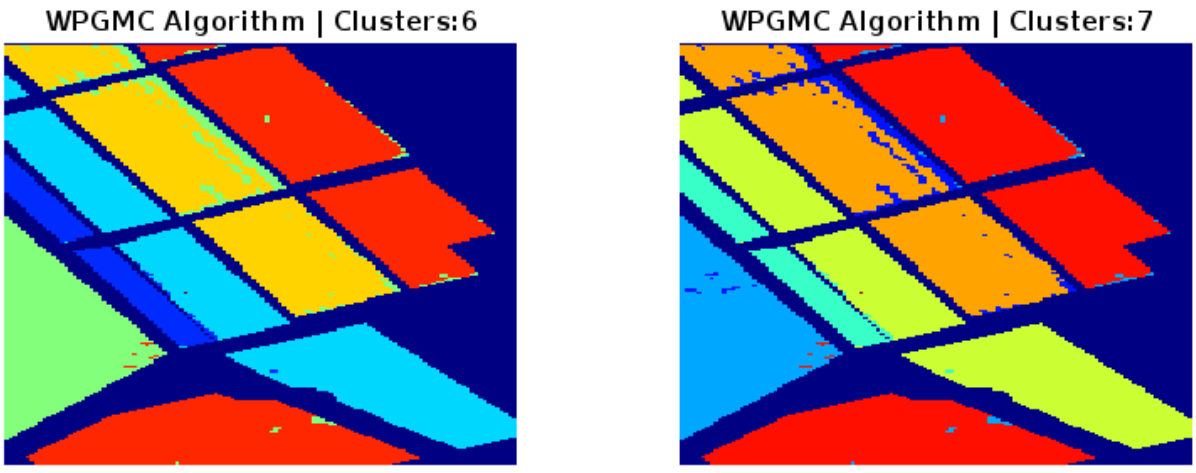
```
% Perform WPGMC Hierarchical Clustering for specified cluster numbers
Total_idx_WPGMC = [];
for i = [6, 7]
    % Using 'median' method in linkage for WPGMC
```

```
    B = linkage(Z, 'median', 'euclidean');
    idx = cluster(B, 'maxclust', i);
    Total_idx_WPGMC = cat(2, Total_idx_WPGMC, idx);
end
```

We then use the following code to perform qualitative evaluation (plot the Ground Truth and the first Principal Component vs the clustering results):

```
% Reshape the first PC back to the original image dimensions for visualization
Z_total_cube_WPGMC = reshape(PC1_reshaped, p, n);

% Plot the Ground Truth, PC1, and WPGMC clustering results
clust_eval(Total_idx_WPGMC, 'WPGMC Algorithm', Z_total_cube_WPGMC, salinas_gt, existed_L
    );
```





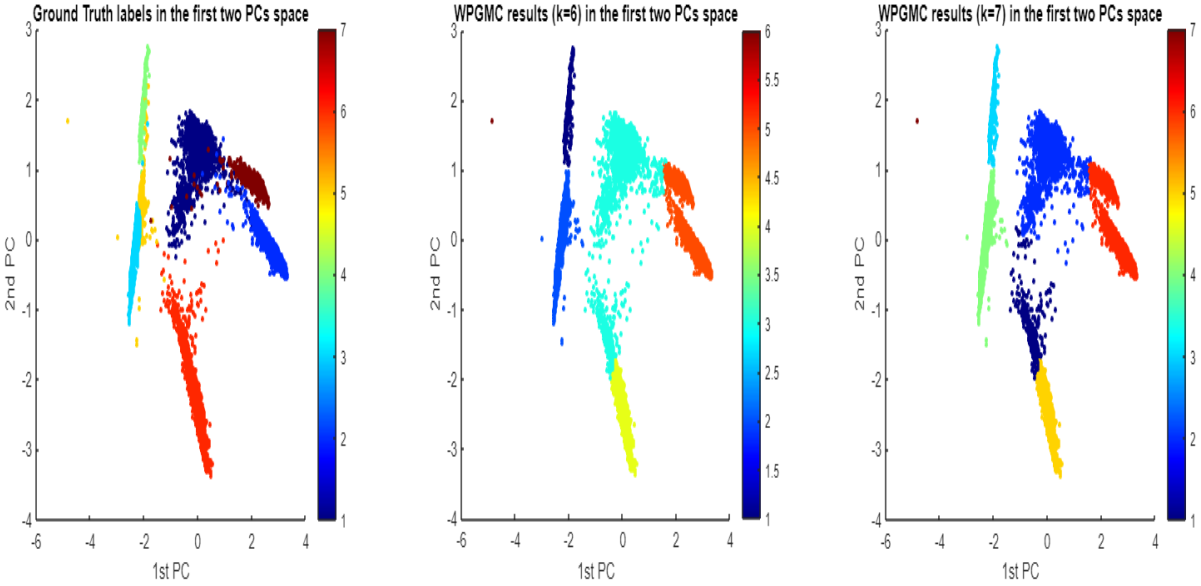
The non-monotonic nature of the cluster tree in WPGMC results in a disjointed and inconsistent clustering structure. This is evident in the image where the algorithm fails to appropriately segregate and match the ground truth clusters, particularly in the 6-cluster scenario. With 7 clusters, the over-segmentation tends to be somewhat mitigated but still does not capture the true distribution of the data, leading to a less than satisfactory representation of the actual classes present in the ground truth.

WPGMC led to a non-monotonic hierarchical clustering, where certain cluster combinations did not progressively increase in similarity as expected. This is observed when a clustering step combines clusters at a distance less than the maximum distance of individual data points within the clusters (crossover), defying the expected pattern of a monotonic increase in similarity seen in methods like Complete Link and Ward's. Consequently, this irregularity in clustering hierarchy adversely affects the performance of WPGMC for both 6 and 7 clusters, resulting in a clustering outcome that does not align

well with either the first Principal Component or the ground truth.

To plot the WPGMC clustering results vs the Ground truth PCA in the space of the first 2 PCs, we use the following code:

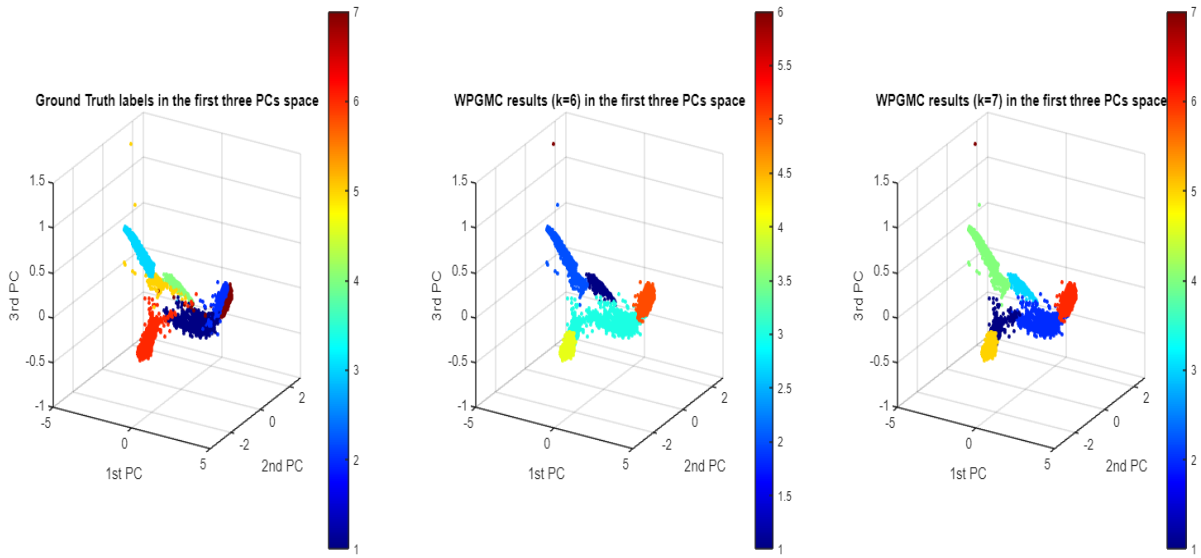
```
if DISPLAY_FIGURES
    % Set the figure size for 2D plots
    figure('Position', [100, 100, 1200, 300]);

    % Ground truth labels for the first two principal components
    subplot(1, 3, 1);
    scatter(Z(:,1), Z(:,2), 8, Ground_truth, 'filled', 'MarkerEdgeAlpha', 0.9);
    title('Ground Truth labels in the first two PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    colormap('jet');
    colorbar;
    axis square;

    % WPGMC clustering results for k=6 in the first two principal components
    subplot(1, 3, 2);
    scatter(Z(:,1), Z(:,2), 8, Total_idx_WPGMC(:,1), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('WPGMC results (k=6) in the first two PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    colormap('jet');
    colorbar;
    axis square;

    % WPGMC clustering results for k=7 in the first two principal components
    subplot(1, 3, 3);
    scatter(Z(:,1), Z(:,2), 8, Total_idx_WPGMC(:,2), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('WPGMC results (k=7) in the first two PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    colormap('jet');
    colorbar;
    axis square;
end
```



It is evident that the WPGMC algorithm, while applied to six and seven clusters, does not align optimally with the ground truth. Specifically, the results for k=6 show a significant divergence from the actual class distributions, with substantial overlap between clusters. Moving to k=7, although there is a slight improvement in cluster separation, the algorithm still does not accurately capture the distinct groups as outlined in the ground truth. The non-monotonic nature of the cluster tree contributes to these discrepancies, impacting the algorithm's ability to consistently merge clusters in a way that reflects the true underlying structure of the data.

To plot the WPGMC clustering results vs the Ground truth PCA in the space of the first 3 PCs, we use the following code:

```
if DISPLAY_FIGURES
    % Set the figure size for 3D plots
    figure('Position', [100, 100, 1200, 400]);

    % Ground truth labels for the first three principal components
    subplot(1, 3, 1);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Ground_truth, 'filled');
    title('Ground Truth labels in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]);

    % WPGMC clustering results for k=6 in the first three principal components
    subplot(1, 3, 2);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Total_idx_WPGMC(:,1), 'filled');
    title('WPGMC results (k=6) in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]);

    % WPGMC clustering results for k=7 in the first three principal components
    subplot(1, 3, 3);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Total_idx_WPGMC(:,2), 'filled');
    title('WPGMC results (k=7) in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]);
end
```



For k=6, clusters seem merged and not well-defined, while for k=7, the algorithm appears to create clusters that are slightly more separated but still not entirely congruent with the ground truth labels. This suggests that the non-monotonic nature of the cluster tree hinders the algorithm's ability to capture the true complexity and separation of the data, leading to suboptimal clustering performance.

4.2.3 Ward's Algorithm

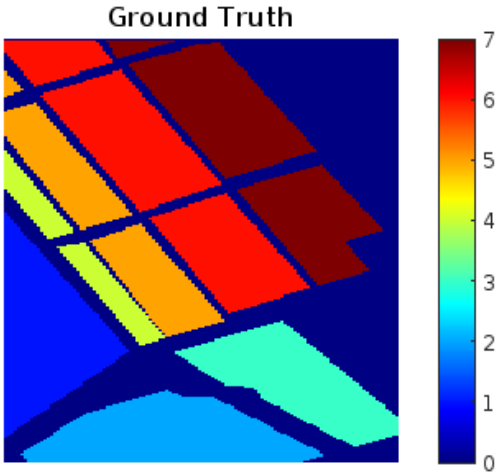
The Ward clustering method emphasizes minimizing within-cluster variance rather than relying solely on the distance between data points. It assesses the impact of combining clusters by measuring the increase in the total within-cluster sum of squares, aiming to join clusters with minimal increase in total variance. This approach yields clusters that are comparably similar within themselves.

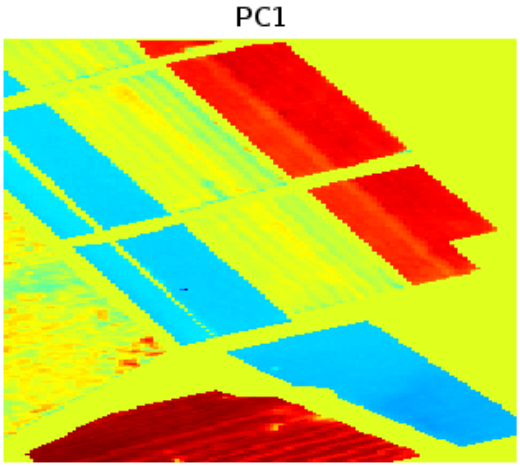
We used the following code:

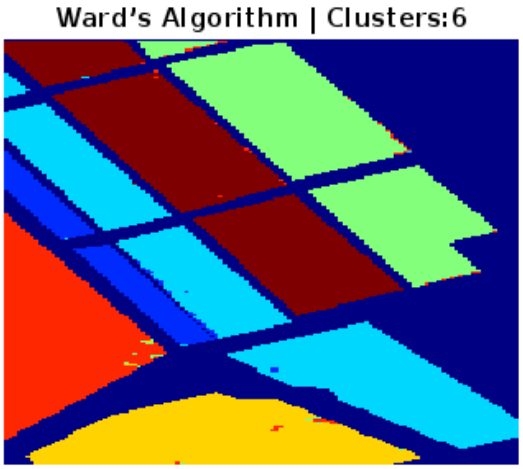
```
% Perform Ward's Hierarchical Clustering for specified cluster numbers
Total_idx_Ward = [];
for i = [6, 7]
    B = linkage(Z, 'ward', 'euclidean');
    idx = cluster(B, 'maxclust', i);
    Total_idx_Ward = cat(2, Total_idx_Ward, idx);
end
```

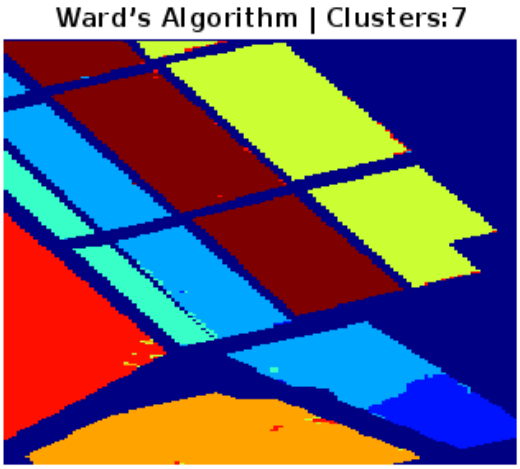
We then use the following code to perform qualitative evaluation (plot the Ground Truth and the first Principal Component vs the clustering results):

```
% Reshape the first PC back to the original image dimensions for visualization
Z_total_cube_Ward = reshape(PC1_reshaped, p, n);

% Plot the Ground Truth, PC1, and Ward's clustering results
clust_eval(Total_idx_Ward, 'Ward Algorithm', Z_total_cube_Ward, salinas_gt, existed_L);
```







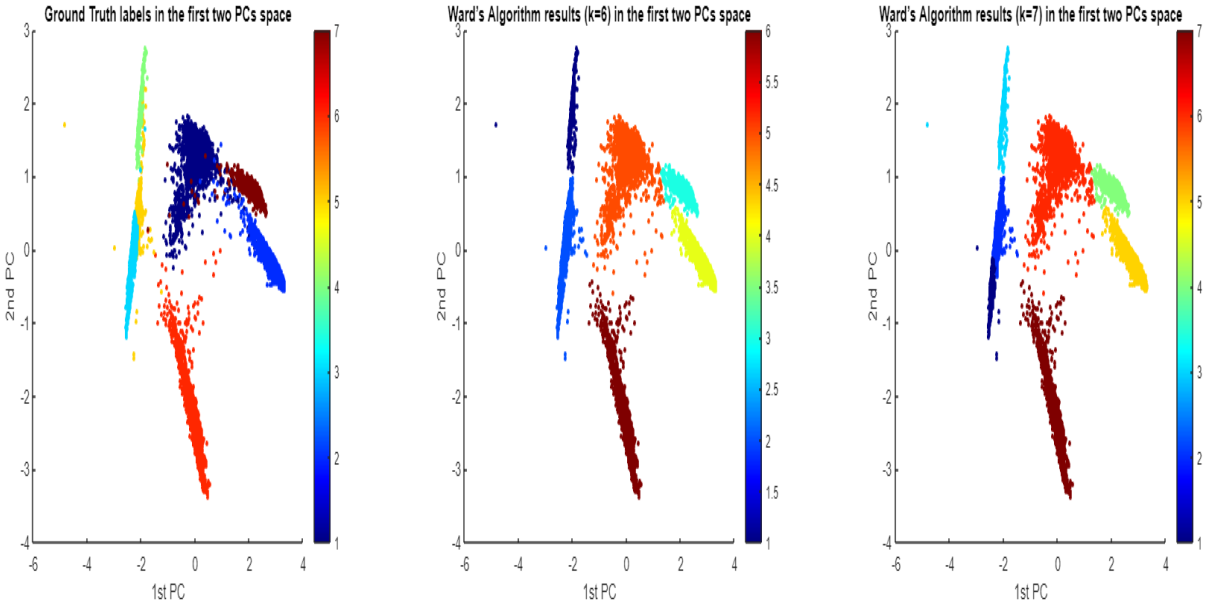


Ward's approach is designed to minimize the total within-cluster variance, leading to a segmentation that mirrors the delineation offered by PC1, particularly for six clusters. However, while Ward's method

appears to effectively capture the data's structure, the comparison to the ground truth reveals a recurrent issue—the conflation of labels 3 and 5 into a single cluster. This suggests that while Ward's algorithm excels in identifying the broad structure of the data, it struggles to fully distinguish between closely related classes, an issue that persists even as the number of clusters is increased to seven.

To plot the Ward clustering results vs the Ground truth PCA in the space of the first 2 PCs, we use the following code:

```
% Plot ground truth labels and Ward's clustering results for the first two principal
    components
if DISPLAY_FIGURES
    % Set the figure size for 2D plots
    figure('Position', [100, 100, 1200, 300]);

    % Ground truth labels for the first two principal components
    subplot('Position', [0.05, 0.1, 0.28, 0.8]);
    scatter(Z(:,1), Z(:,2), 8, Ground_truth, 'filled', 'MarkerEdgeAlpha', 0.9);
    title('Ground Truth labels in the first two PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    colormap('jet');
    colorbar;
    axis square;

    % Ward's clustering results for k=6 in the first two principal components
    subplot('Position', [0.37, 0.1, 0.28, 0.8]);
    scatter(Z(:,1), Z(:,2), 8, Total_idx_Ward(:,1), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('Ward Algorithm results (k=6) in the first two PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    colormap('jet');
    colorbar;
    axis square;

    % Ward's clustering results for k=7 in the first two principal components
    subplot('Position', [0.69, 0.1, 0.28, 0.8]);
    scatter(Z(:,1), Z(:,2), 8, Total_idx_Ward(:,2), 'filled', 'MarkerEdgeAlpha', 0.9);
    title('Ward Algorithm results (k=7) in the first two PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    colormap('jet');
    colorbar;
    axis square;
end
```

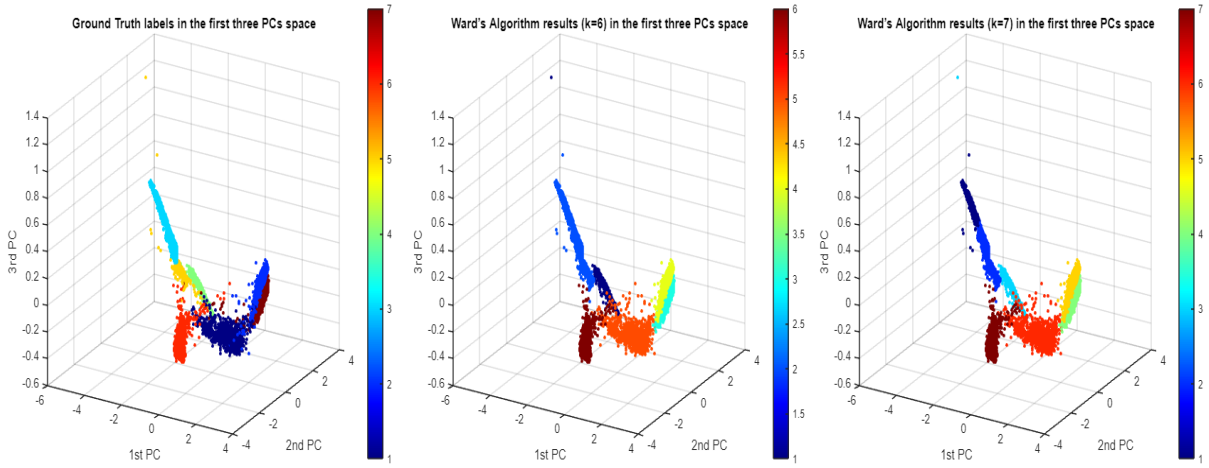


Ward's algorithm results for k=6 show a clear but not perfect partitioning of the data into clusters, with some minor overlap and potential under-segmentation issues. As the number of clusters increases to

k=7, the algorithm begins to over-segment, creating additional clusters which may not correspond to actual distinct classes in the ground truth.

To plot the Ward clustering results vs the Ground truth PCA in the space of the first 3 PCs, we use the following code:

```
% Plot ground truth labels and Ward's clustering results for the first three principal
    components
if DISPLAY_FIGURES
    % Set the figure size for 3D plots
    figure('Position', [100, 100, 1200, 400]);

    % Ground truth labels for the first three principal components
    subplot('Position', [0.05, 0.1, 0.28, 0.8]);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Ground_truth, 'filled');
    title('Ground Truth labels in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]);

    % Ward's clustering results for k=6 in the first three principal components
    subplot('Position', [0.37, 0.1, 0.28, 0.8]);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Total_idx_Ward(:,1), 'filled');
    title('Ward's Algorithm results (k=6) in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]);

    % Ward's clustering results for k=7 in the first three principal components
    subplot('Position', [0.69, 0.1, 0.28, 0.8]);
    scatter3(Z(:,1), Z(:,2), Z(:,3), 8, Total_idx_Ward(:,2), 'filled');
    title('Ward's Algorithm results (k=7) in the first three PCs space');
    xlabel('1st PC');
    ylabel('2nd PC');
    zlabel('3rd PC');
    colormap('jet');
    colorbar;
    axis square;
    view([30 20]);
end
```



With six clusters, Ward's algorithm manages to identify several distinct groups, though it appears to under-segment the data, merging some separate classes. Transitioning to seven clusters, the algorithm demonstrates a tendency towards over-segmentation, fragmenting what might be coherent clusters into smaller parts. This is what we observed before in the 2 PCs space, shown even clearer.

5 Quantitative Evaluation of the Results

The Rand index is a measure of the similarity between two data clusterings. Specifically, it is a way to quantify how well a clustering algorithm has performed compared to some ground truth, often in the form of human-annotated labels.

Here is how the Rand index works:

1. **True Positives (TP)**: The number of pairs of elements that are in the same cluster in both the ground truth and the clustering result
2. **True Negatives (TN)**: The number of pairs of elements that are in different clusters in both the ground truth and the clustering result
3. **False Positives (FP)**: The number of pairs of elements that are in the same cluster in the clustering result but in different clusters in the ground truth
4. **False Negatives (FN)**: The number of pairs of elements that are in the same cluster in the ground truth but in different clusters in the clustering result

The Rand index is calculated using the formula:

$$\text{Rand Index} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

This formula essentially measures the fraction of correct decisions made by the clustering algorithm. The Rand index ranges from 0 to 1, where 1 indicates perfect clustering alignment with the ground truth and 0 indicates no alignment.

The Rand index is a great choice for evaluating the clustering results in this project, as it offers several benefits. It enables a direct and clear comparison of the clustering outcomes with the ground truth, making it an effective tool for assessing performance across the different algorithms. Its range from 0 to 1, where higher values indicate better performance, makes it intuitively easy to understand. Importantly, the Rand index is effective in handling imbalanced data, a common scenario in real-world datasets, and is non-parametric, meaning it does not assume any specific data distribution. This makes it a versatile and objective method to assess the effectiveness of various clustering strategies on the Salinas hyperspectral image dataset.

5.1 Cluster Validation using the Rand Index

We use the following code:

```
% Evaluate k-means clustering results using Rand Index for k=6
randIndex_k6 = rand_index(Total_idx_kmeans(:,1), Ground_truth);
disp(['Rand Index for k-means with k=6: ', num2str(randIndex_k6)]);

% Evaluate k-means clustering results using Rand Index for k=7
randIndex_k7 = rand_index(Total_idx_kmeans(:,2), Ground_truth);
disp(['Rand Index for k-means with k=7: ', num2str(randIndex_k7)]);

% Evaluate FCM clustering results using Rand Index for k=6
randIndex_FCM_k6 = rand_index(Total_idx_FCM(:,1), Ground_truth);
disp(['Rand Index for Fuzzy C-Means with k=6: ', num2str(randIndex_FCM_k6)]);

% Evaluate FCM clustering results using Rand Index for k=7
randIndex_FCM_k7 = rand_index(Total_idx_FCM(:,2), Ground_truth);
disp(['Rand Index for Fuzzy C-Means with k=7: ', num2str(randIndex_FCM_k7)]);

% Evaluate PCM clustering results using Rand Index for k=6
randIndex_PCM_k6 = rand_index(Total_idx_PCM(:,1), Ground_truth);
disp(['Rand Index for Possibilistic C-Means with k=6: ', num2str(randIndex_PCM_k6)]);

% Evaluate PCM clustering results using Rand Index for k=7
randIndex_PCM_k7 = rand_index(Total_idx_PCM(:,2), Ground_truth);
disp(['Rand Index for Possibilistic C-Means with k=7: ', num2str(randIndex_PCM_k7)]);

% Evaluate PBCM clustering results using Rand Index for k=6
randIndex_PBCM_k6 = rand_index(Total_idx_PBCM(:,1), Ground_truth);
```

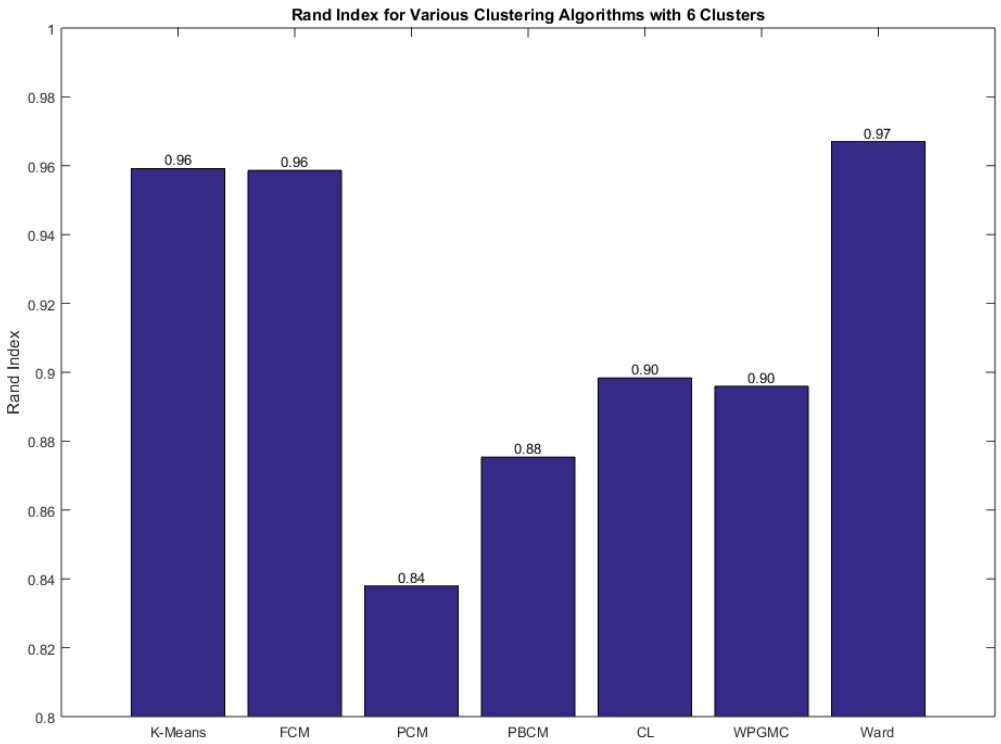
```
disp(['Rand Index for Probabilistic C-Means with k=6: ', num2str(randIndex_PBCM_k6)]);

% Evaluate PBCM clustering results using Rand Index for k=7
randIndex_PBCM_k7 = rand_index(Total_idx_PBCM(:,2), Ground_truth);
disp(['Rand Index for Probabilistic C-Means with k=7: ', num2str(randIndex_PBCM_k7)]);

% Evaluate Complete-Link clustering results using Rand Index for k=6
randIndex_CL_k6 = rand_index(Total_idx_CL(:,1), Ground_truth);
disp(['Rand Index for Complete-Link with k=6: ', num2str(randIndex_CL_k6)]);

% Evaluate Complete-Link clustering results using Rand Index for k=7
randIndex_CL_k7 = rand_index(Total_idx_CL(:,2), Ground_truth);
disp(['Rand Index for Complete-Link with k=7: ', num2str(randIndex_CL_k7)]);

% Evaluate WPGMC clustering results using Rand Index for k=6
randIndex_WPGMC_k6 = rand_index(Total_idx_WPGMC(:,1), Ground_truth);
disp(['Rand Index for WPGMC with k=6: ', num2str(randIndex_WPGMC_k6)]);

% Evaluate WPGMC clustering results using Rand Index for k=7
randIndex_WPGMC_k7 = rand_index(Total_idx_WPGMC(:,2), Ground_truth);
disp(['Rand Index for WPGMC with k=7: ', num2str(randIndex_WPGMC_k7)]);

% Evaluate Ward's clustering results using Rand Index for k=6
randIndex_Ward_k6 = rand_index(Total_idx_Ward(:,1), Ground_truth);
disp(['Rand Index for Ward Algorithm with k=6: ', num2str(randIndex_Ward_k6)]);

% Evaluate Ward's clustering results using Rand Index for k=7
randIndex_Ward_k7 = rand_index(Total_idx_Ward(:,2), Ground_truth);
disp(['Rand Index for Ward Algorithm with k=7: ', num2str(randIndex_Ward_k7)]);

function RI = rand_index(clustering1, clustering2)
    %RAND_INDEX Computes the Rand index to evaluate clustering performance
    %   RI = rand_index(clustering1, clustering2) returns the Rand index, which is
    %   a measure of the similarity between two data clusterings.

    n = length(clustering1);  % Number of elements

    % Initialize counts
    a = 0; % Pairs clustered together in both clusterings
    b = 0; % Pairs clustered separately in both clusterings

    % Iterate over all pairs of elements
    for i = 1:n-1
        for j = i+1:n
            % Check if the pair is clustered similarly in both clusterings
            if (clustering1(i) == clustering1(j) && clustering2(i) == clustering2(j)) ||
     clustering1(i) ~= clustering1(j) && clustering2(i) ~= clustering2(j))
                a = a + 1;
            else
                b = b + 1;
            end
        end
    end

    % Calculate Rand index
    RI = a / (a + b);
end
```

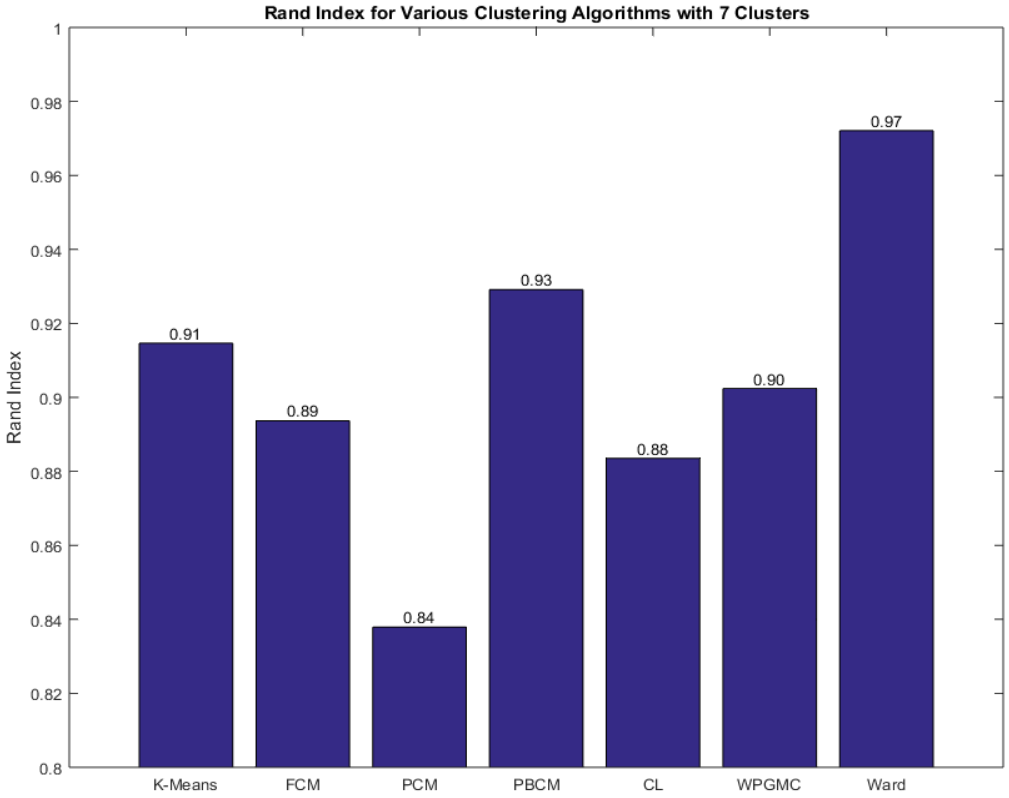
	K-Means	FCM	PCM	PBCM	CL	WPGMC	Ward
6 Clusters	0.95916	0.95862	0.83794	0.87536	0.89835	0.89594	0.96699
7 Clusters	0.91465	0.89371	0.83792	0.92915	0.88354	0.90239	0.97214

Table 1: Rand Index scores comparing the various clustering algorithms to the ground truth for 6 and 7 clusters

To better visualize these results, we use the following code:

```
% Data for Rand Index values
algorithms = {'K-Means', 'FCM', 'PCM', 'PBCM', 'CL', 'WPGMC', 'Ward'};
randIndex_6Clusters = [randIndex_k6, randIndex_FCM_k6, randIndex_PCM_k6,
    randIndex_PBCM_k6, randIndex_CL_k6, randIndex_WPGMC_k6, randIndex_Ward_k6];
randIndex_7Clusters = [randIndex_k7, randIndex_FCM_k7, randIndex_PCM_k7,
    randIndex_PBCM_k7, randIndex_CL_k7, randIndex_WPGMC_k7, randIndex_Ward_k7];

% Bar plot for 6 clusters
figure;
bar(randIndex_6Clusters);
set(gca, 'xticklabel', algorithms);
ylabel('Rand Index');
title('Rand Index for Various Clustering Algorithms with 6 Clusters');
ylim([0.8 1]); % Assuming the Rand Index values are between 0.8 and 1

% Adding the values on top of the bars
for i = 1:length(randIndex_6Clusters)
    text(i, randIndex_6Clusters(i), num2str(randIndex_6Clusters(i), '%0.2f'), '
    HorizontalAlignment', 'center', 'VerticalAlignment', 'bottom')
end

% Bar plot for 7 clusters
figure;
bar(randIndex_7Clusters);
set(gca, 'xticklabel', algorithms);
ylabel('Rand Index');
title('Rand Index for Various Clustering Algorithms with 7 Clusters');
ylim([0.8 1]); % Adjust the limits based on your data range

% Adding the values on top of the bars
for i = 1:length(randIndex_7Clusters)
    text(i, randIndex_7Clusters(i), num2str(randIndex_7Clusters(i), '%0.2f'), '
    HorizontalAlignment', 'center', 'VerticalAlignment', 'bottom')
end
```





5.2 Discussion

Concerning the performance of each algorithm:

- **K-Means**: Exhibited strong performance with higher Rand Index scores, indicating good alignment with the ground truth. Its simplicity and efficiency in partitioning data into distinct clusters were evident. However, being sensitive to initial conditions, different runs might yield varied results.
- **Fuzzy C-Means (FCM)**: Demonstrated close performance to K-Means. Its strength lies in assigning probabilities to each data point for belonging to different clusters, which can be more representative for certain datasets. However, its tendency to create spherical clusters might not be suitable for all data shapes.
- **Possibilistic C-Means (PCM)**: Scored lower in comparison, possibly due to its approach that allows points to belong to multiple clusters with varying degrees. This can lead to less defined cluster boundaries, impacting performance metrics.
- **Probabilistic C-Means (PBCM)**: Showed intermediate performance. Its probabilistic approach provides flexibility but might struggle with complex cluster shapes or overlapping distributions.
- **Hierarchical Clustering (Complete Link, WPGMC, Ward)**: These algorithms showed varied performance. Complete Link tends to create smaller, more compact clusters, while Ward's method focuses on minimizing variance, leading to more balanced clusters. WPGMC's flexibility didn't align well with the data structure, leading to lower performance.

Ward's algorithm of the Hierarchical Clustering had the best overall performance, particularly with seven clusters. This success could be attributed to its approach of minimizing within-cluster variance, leading to more naturally formed clusters that closely resembled the actual data distribution. It effectively balanced the granularity of cluster formation without over-segmenting, a common challenge in other methods.

Some potential improvement suggestions are:

- Better initialization strategies for K-Means and FCM could enhance performance.
- For PCM and PBCM, refining the way probabilities and memberships are assigned could improve cluster definition.
- Exploring different distance metrics or linkage criteria in hierarchical methods might yield better alignment with certain datasets.

Overall, this project successfully demonstrated the applicability and nuances of various clustering algorithms. The distinct approaches of these algorithms to group data highlight the importance of understanding the underlying data structure and the specific strengths of each algorithm. The best choice of algorithm depends on the specific characteristics of the dataset and the desired outcome of the clustering process. The Ward's method, with its focus on minimizing variance within clusters, proved to be most effective for this particular dataset, underlining the importance of algorithm selection in machine learning tasks.

References

[1] Salinas HSI Dataset. `https://paperswithcode.com/dataset/salinas`.

[2] Visible Light - NASA Science. `https://science.nasa.gov/ems/09_visiblelight/`.

[3] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, April 2016.

[4] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, November 1987.

[5] Armin Schneider and Hubertus Feussner. Chapter 5 - diagnostic procedures. In Armin Schneider and Hubertus Feussner, editors, *Biomedical Engineering in Gastrointestinal Surgery*, pages 87–220. Academic Press, 2017.