Project

Michael Darmanis (7115152200004) `mdarm@di.uoa.gr`

Vasilios Venieris (7115152200017) `vvenieris@di.uoa.gr`

January 26, 2023

1 Introduction

This project uses the data provided in the file `Salinas_Data.mat`, which contains a $150x150x204$ three-dimensional matrix named `Salinas_Image` (representing the Salinas hypercube) and a $150x150$ two-dimensional image named `Salinas_Labels` (representing the class label for each pixel). The aim of this project is to compare the performance of different clustering algorithms in finding homogeneous regions within the Salinas HSI.

The objective is to compare the performance of cost function optimisation clustering algorithms (e.g. k-means, fuzzy c-means, possibilistic c-means and probabilistic clustering) and hierarchical algorithms (e.g. complete-link, WPGMC and Ward algorithms) in identifying homogeneous regions within the Salinas hyperspectral image. The analysis will be restricted to those pixels for which class label information is available.

To achieve this goal, the following tasks will be executed:

- Execution of all above algorithms, for various combinations of their parameters (e.g. the number of clusters etc.) in order to identify the homogeneous regions in the image, and reporting any encountered problems.
- Qualitative verification of the results obtained by each algorithm in terms of (i) the pixel label-information and (ii) information that can be gathered by the image (.i.e examining principal components).
- Quantitative verification of the results in terms of the labelling information.
- Comparison of the overall clustering accuracy for each algorithm.

The aim of the project is to gain insight into how different clustering algorithms perform on the Salinas HSI and to identify which algorithm is best suited to this type of data. Although not explicitly stated, extensive use was made of the following textbooks Pattern Recognition[4] and Mining Massive Datasets[2].

2 Methodology

Before clustering algorithms are applied, a two-step process is used in which: (i) the dimensionality of the data is reduced, and then (i) k-means clustering is performed in the low-dimensional feature space to determine the optimal number of clusters. PCA followed by k-means performs linear dimensionality reduction on the input image to be clustered.

2.1 Hyperspectral dataset

More specifically, after loading the data, each feature of the dataset is normalised with respect to its mean and its range. Principal component analysis (PCA) is then applied to the normalised data to determine the optimal number of components to retain, while still retaining at least 99% of the variance in the data set. However, variance explainability does not necessarily imply reconstructability. This is clearly shown in the `biplot` [6] of Figure 1.

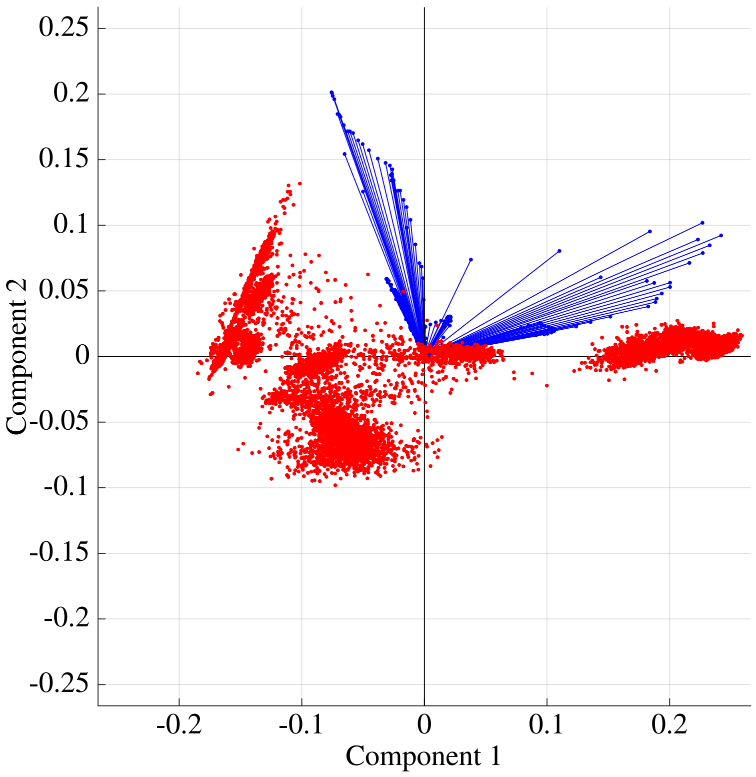


Figure 1: Principal components retaining 99% of the initial variance.

While the two principal components retain more than 99% of the data (red dots), the vectors (blue line segments) have a large angle, indicating a loss in reconstruction. However, this worst case is a necessity. Due to the high dimensionality of the data (204 bands), PCA is performed for reasons of computational efficiency, while allowing a more heuristic and less rigorous analysis.

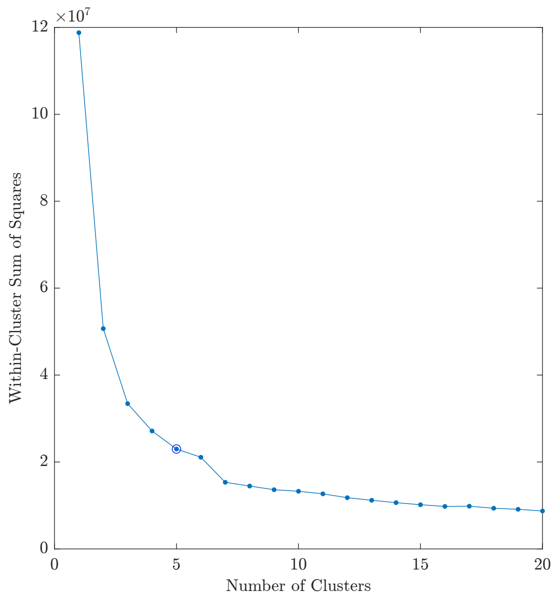
The Calinski-Harabasz criterion and the ”elbow” method are then used to determine the optimal number of clusters, as can be seen in Figure 2. The optimal number of clusters was found to be

```
Enter the number of clusters k (Calinski-Harabasz criterion = 7, Elbow Method = 5):
```

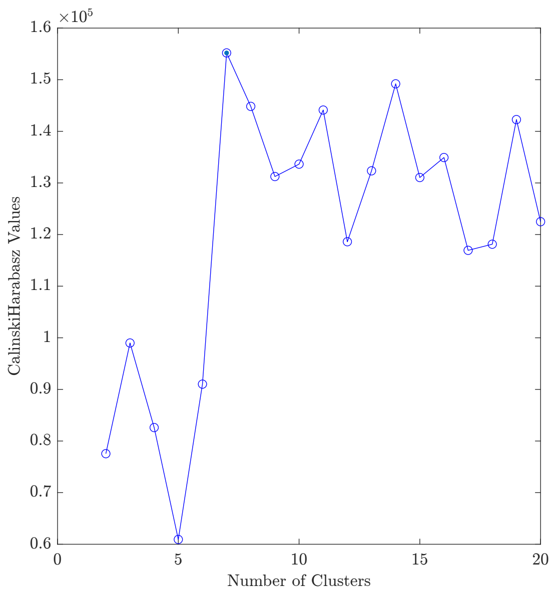
It should be noted that the “elbow” method is a heuristic method used to determine the number of clusters, by fitting the model with a range of number of clusters and looking for an “elbow” point on the within-cluster sum of squares (WCSS) plot, where the increase in WCSS slows down. The fallacious indication of 5 number of clusters is a result of an attempted automation of the process using the second derivative as can be seen in the following part of the code:

```
% Calculate the second derivative of the WCSS
d2 = diff(diff(wcss));

% Find the local maxima of the second derivative
[~, idx] = findpeaks(-d2);

% The optimal number of clusters is the number of clusters where the second
% derivative changes sign
k1 = idx(1) + 2;
```

The analysis will thus proceed with 7 number of clusters.



(a) "Elbow" method.



(b) Variance-ratio criterion.

Figure 2: Picking the right number of initial clusters k.

2.2 Implementation

After pre-processing and dimensionality reduction, the modified data is then subjected to a comprehensive examination using a variety of clustering algorithms. These algorithms include both cost function optimisation techniques and hierarchical approaches. Specifically, the cost function optimisation algorithms used in this analysis include k-means, fuzzy c-means, possibilistic c-means and the probabilistic algorithm. Hierarchical algorithms such as complete linkage, weighted pair group method with arithmetic mean (WPGMA) and ward linkage are also used.

The hierarchical algorithms are applied to the same dataset as the cost function optimisation algorithms and the results obtained are compared with those obtained using the original label figure. The subsequent analysis is both qualitative and quantitative, with particular emphasis on assessing the overall clustering accuracy of the algorithms employed.

3 Experiments and results

Most of the clustering algorithms used in this analysis are run while varying a number of their parameters. The optimal run, determined by a specific criterion, is then retained for comparison. The reconstructed images based on the clustered labels are presented using principal component analysis. In the case of hierarchical algorithms, dendrograms are also presented for clarity. Finally, the best results of each algorithm are evaluated using the Adjusted Rank Index (ARI) [3] and the Normalised Mutual Information (NMI) [1] as performance metrics.

3.1 Cost-function-optimisation algorithms

The k-means algorithm is used first, where it is run multiple times to select the iteration with the lowest sum of distances (sumd) between the data points and their corresponding cluster centroids. It was found that a modification of the metric distance was necessary due to the presence of noise in the data, so its variant (the k-medians) was actually executed. The resulting cluster assignments are then used to assign class labels to each pixel.

The fuzzy c-means algorithm is then used, with different values for the fuzzifier parameter, and the iteration with the lowest cost function value is selected. The resulting cluster assignments are then used to assign class labels to each pixel.

The Possibilistic c-means algorithm is then used with a fixed eta, a q-number of 40 and a random initialisation of centroids to assign class labels to each pixel. No attempt was made to automate the process. Parameter variation was done manually and the best quantitative result was kept.

Finally, a probabilistic algorithm is used to fit a Gaussian mixture model to the data for different values of k, considering two types of covariance (a full covariance matrix and a diagonal). The model with the lowest Bayesian Information Criterion (BIC) [5] value is selected and the resulting cluster assignments are used to assign class labels to each pixel.

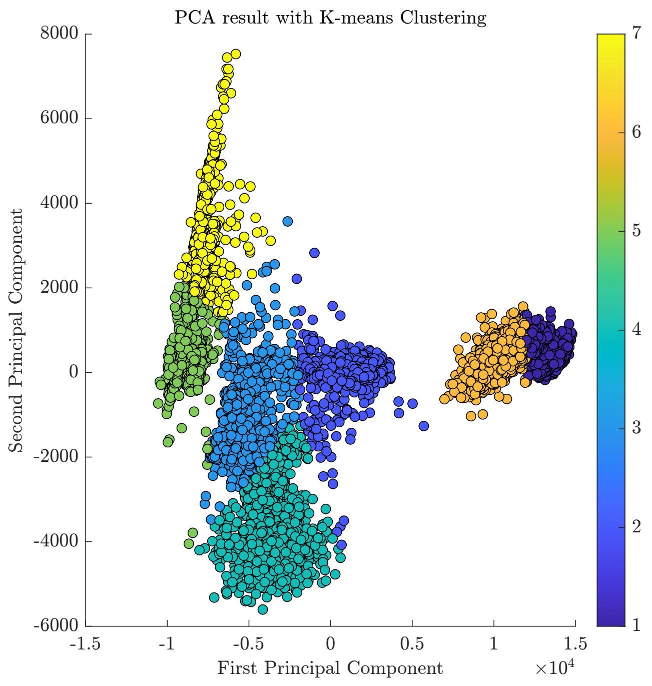


Figure 3: Principal components on the resulting k-means clustering.

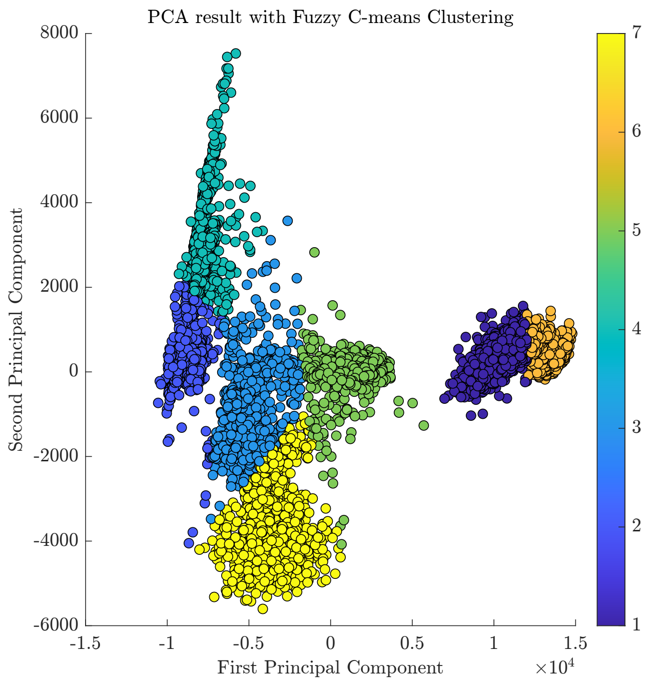


Figure 4: Principal components on the resulting fuzzy c-means clustering.

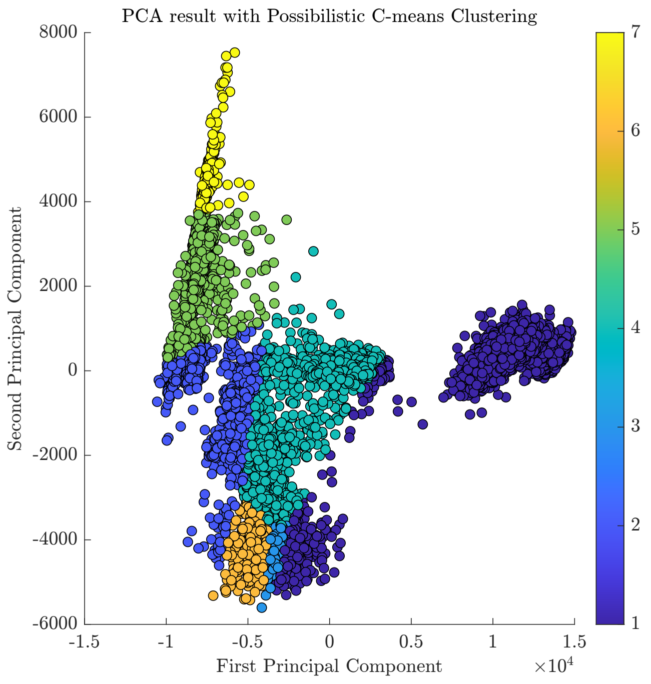


Figure 5: Principal components on the resulting possibilistic c-means clustering.

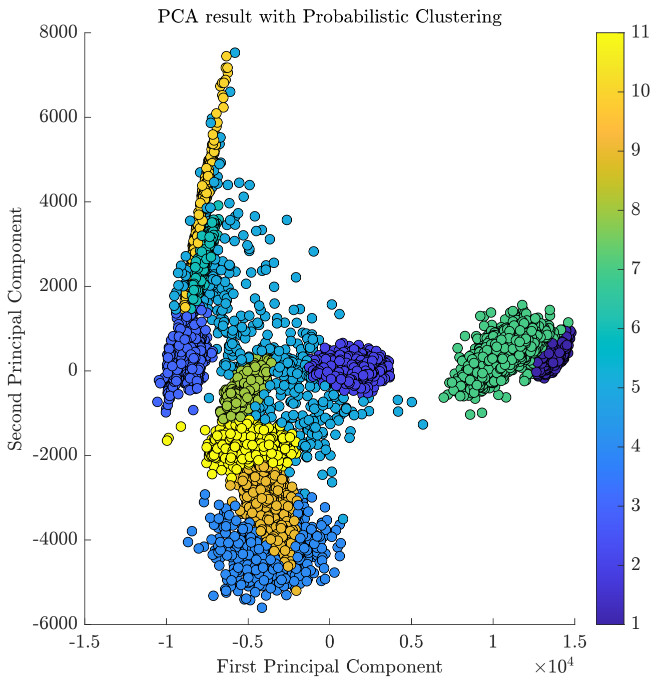


Figure 6: Principal components on the resulting probabilistic clustering.

3.2 Hierarchical algorithms

Three hierarchical clustering algorithms were applied to the modified data obtained from pre-processing and dimensionality reduction. The complete linkage, weighted pair group method with arithmetic mean (WPGMC) linkage, and Ward linkage algorithms were employed.

The complete linkage algorithm was applied by first computing the pairwise distances between the data points using a pool of dissimilarity metrics, including Euclidean, cityblock (Manhattan), Mahalanobis, and Chebychev. The linkage algorithm was then applied to the resulting distance matrix. The script then calculated the cophenetic correlation coefficient between the resulting linkage matrix and the original distance matrix, and selected the linkage matrix and the corresponding metric that resulted in the highest coefficient. The best linkage matrix and the corresponding metric were then used to assign class labels to each pixel. The distance yielding the best results was the Manhattan one, as the script's output indicates:

```
The best metric, for complete linkage, is cityblock with a cophenetic correlation
coefficient of 0.904063
```

The WPGMC linkage and Ward linkage algorithms were also applied, using only the Euclidean distance to compute pairwise distances. The linkage matrices obtained in this step were visualised using dendrograms to provide further clarification.

It is worth noting that the complete-linkage algorithm permits the use of any proximity measures, and results will depend on the chosen measure. However, for the WPGMC and Ward linkage algorithms, only squared Euclidean distances should be used for the sake of geometric correctness, as these methods compute centroids in Euclidean space.

The termination criterion for all of the above-mentioned algorithms was the initially calculated ideal number of clusters, which was set to 7.

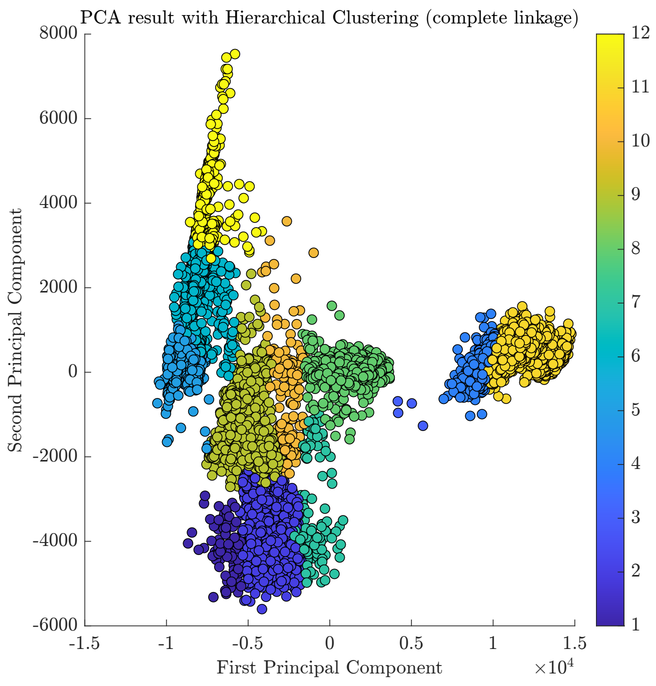


Figure 7: Principal components on the resulting hierarchical clustering with complete linkage.

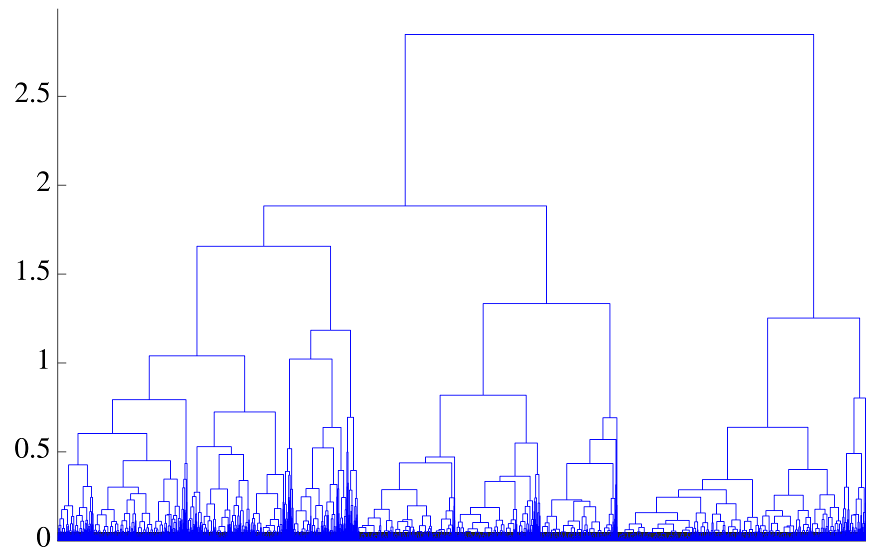


Figure 8: Dendrogram of hierarchical algorithm with complete linkage.

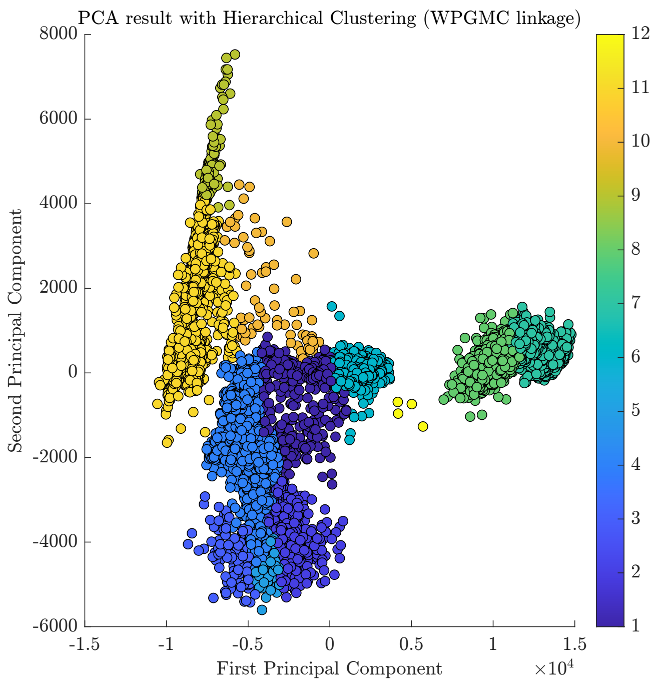


Figure 9: Principal components on the resulting hierarchical clustering with WPGMC.

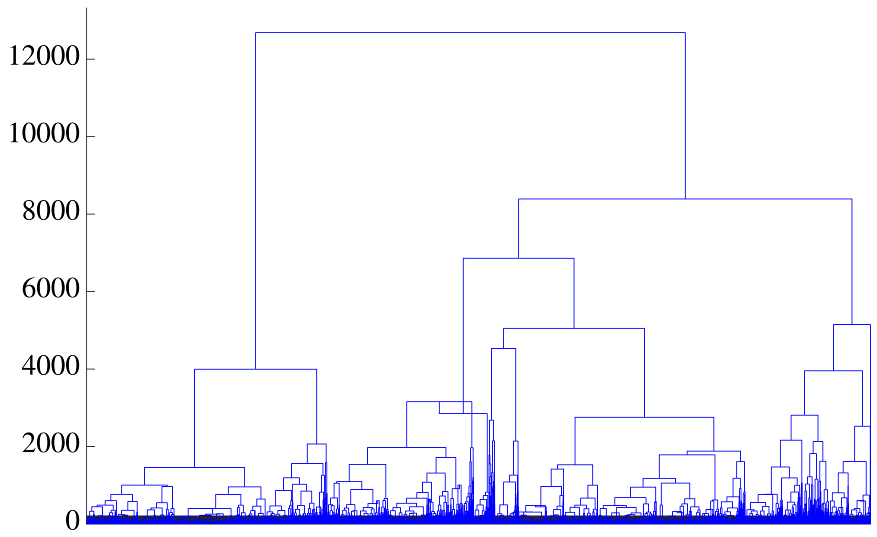


Figure 10: Dendrogram of hierarchical algorithm with WPGMC.

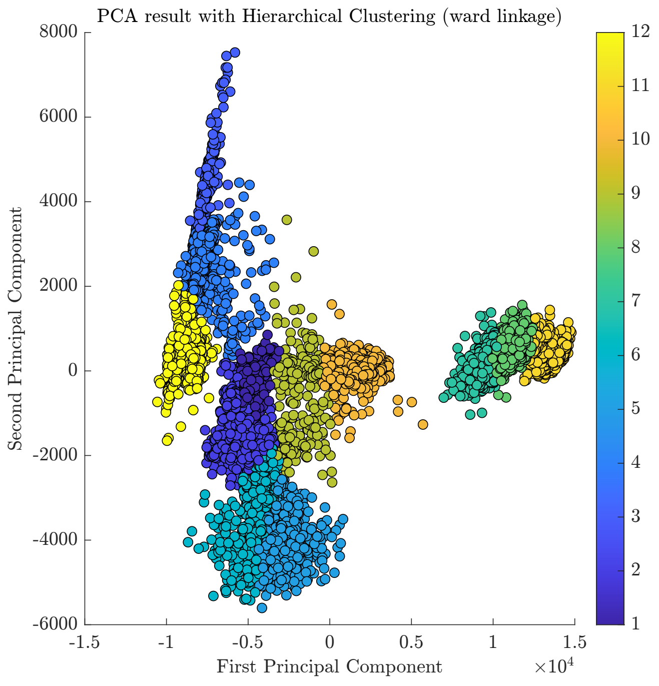


Figure 11: Principal components on the resulting hierarchical clustering with Ward.

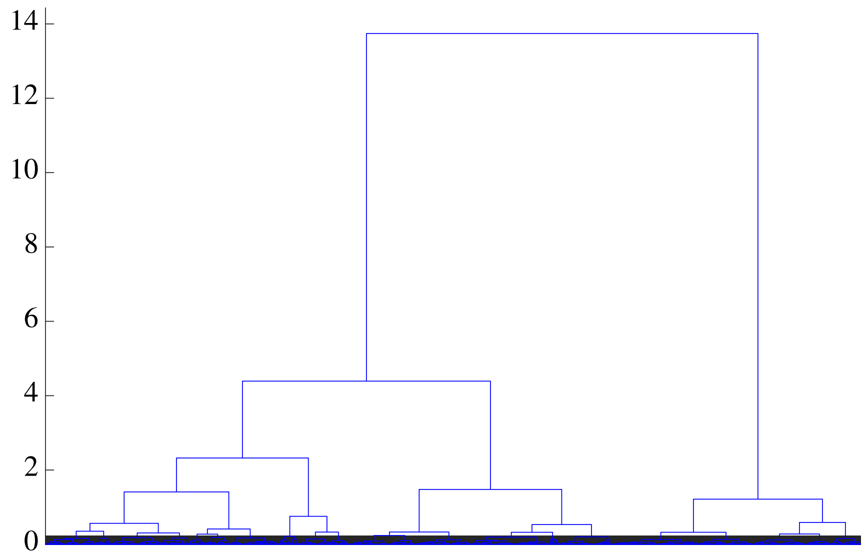


Figure 12: Dendrogram of hierarchical with Ward.

3.3 Qualitative evaluation

Figure 13 showcases the reconstructed figure using the clustered labels, while Figure 14 is a representation of the two principal components of the original figure.

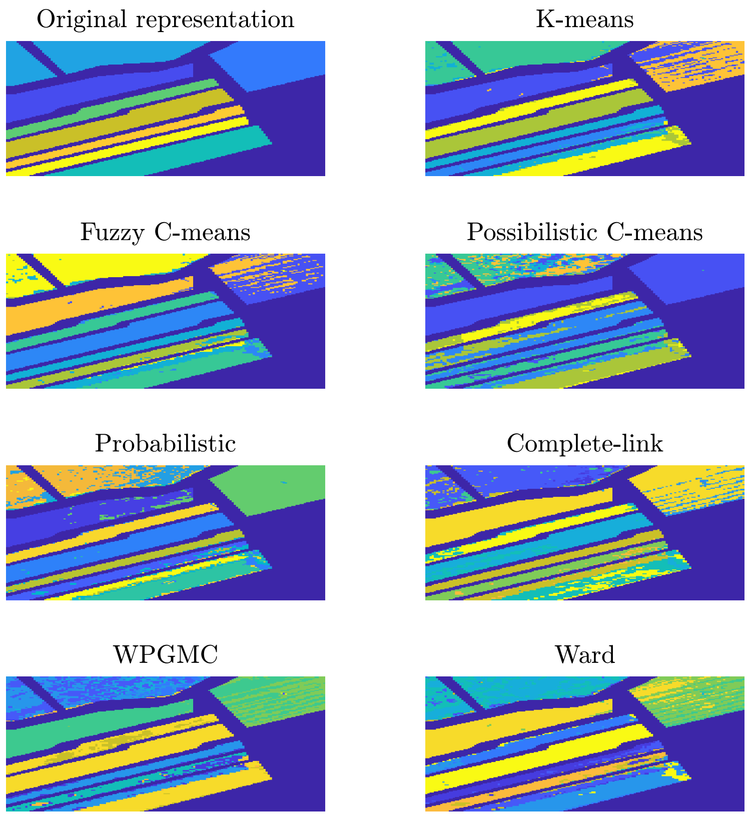


Figure 13: Reconstructed images, using original labels and resultant clustering labels.

The k-means and fuzzy c-means algorithms were found to produce similar results in identifying the underlying structure of a dataset, as shown in Figures 3 and 4. The fuzzy c-means algorithm is an extension of the k-means algorithm that allows for overlapping clusters, but this feature did not prove beneficial in the case study. A limitation of both algorithms is that they assume compact clusters with equal variances and are sensitive to noise. While applying the algorithms several times with different

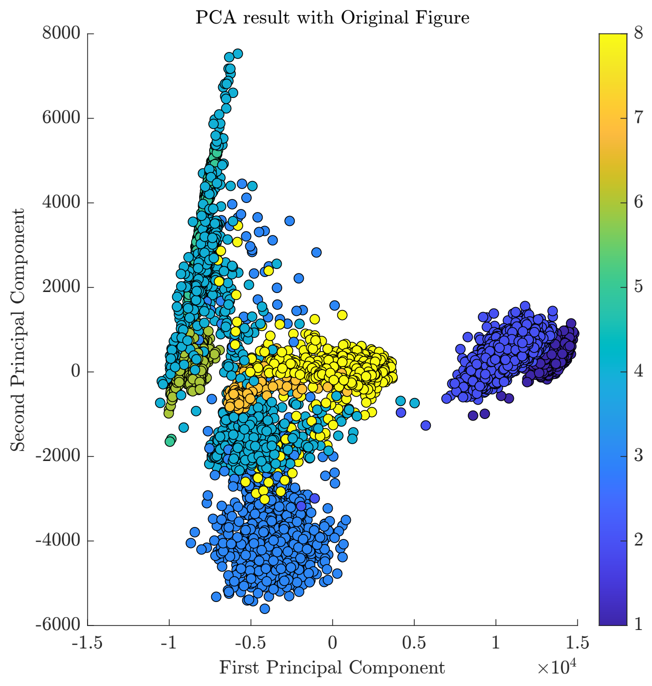


Figure 14: Principal components of the original representation.

initial conditions can mitigate the first drawback, the issue of cluster shape remains unresolved.

The possibilistic c-means algorithm, another extension of the k-means algorithm, allows for overlapping clusters. However, it has been found to produce the worst clustering results among the cost function optimisation algorithms, as shown in Figure 5. This is mainly due to the lack of proper parameterisation and the influence of noise on its efficiency. Theoretically, the probabilistic c-means algorithm could be ideal for the Salinas dataset if properly parameterised.

In contrast, the probabilistic clustering algorithm is based on the assumption that the data follows a Gaussian mixture model. Like the previous algorithms, it is sensitive to initial conditions. This algorithm produced the best result when assuming a diagonal covariance matrix for the clusters, as shown in Figure 6. It was able to capture the nuanced structure of the Salinas dataset. However, it should be noted that the best result corresponded to 11 clusters, which were found to be incorrect by external validation, as shown in Figure 14.

The complete linkage method, as shown in Figure 7, does reveal some clusters, but they are not homogeneous. This is due to the method's metaphor of cluster formation, which is similar to a circle in which the two most distant members cannot be much more dissimilar than other dissimilar pairs. Such clusters are compact at their boundaries, but not necessarily compact inside, and are sensitive to noise.

The WPGMC algorithm also performs poorly at identifying compact clusters, as shown in Figure 9. This is due to the fact that the cluster centroids are defined such that the sub-clusters have an equal influence on their centroid, regardless of the number of objects in each sub-cluster. This method is also sensitive to the initial configuration of the data and is not well suited to dealing with clusters of different sizes or shapes, as it tends to produce clusters of similar size and shape. In addition, its implementation in this case did not converge to a global optimum due to an instance of non-monotonicity.

Finally, Ward's method, as seen in Figures 11 and 3, is the closest to k-means clustering in terms of properties and efficiency. Both methods have the same objective function, which is to minimise the sum of squares pooled within the cluster. While k-means, given appropriate initial centroids, is a better minimiser of this function, Ward's method has been found to be more accurate in detecting clusters of uneven physical size or clusters that are irregularly distributed in space.

3.3.1 Hierarchical algorithms' miscluster

In the present study, a cut-off of seven clusters was specified for the implementation of all hierarchical clustering algorithms. However, it was observed that all algorithms returned twelve clusters instead.

One possible explanation for this discrepancy may be that the cut-off of seven clusters was not appropriate for the data and the specific hierarchical algorithms employed. The determination of the number of clusters is often based on domain knowledge or heuristics, however, it may also be estimated through the utilization of methods such as the elbow method or the silhouette score. Thus, it is plausible that the chosen cut-off of seven clusters did not accurately reflect the true underlying structure of the data.

Another potential explanation for this discrepancy is that the hierarchical algorithms utilized in this project tend to favour larger numbers of clusters and may not have been able to effectively identify the desired number of clusters. Hierarchical algorithms typically rely on linkage criteria which can be sensitive to the presence of noise or outliers in the data, which can lead to the formation of additional clusters. Additionally, the linkage criteria and dissimilarity measures employed by the hierarchical algorithms may have been less suited to the specific characteristics of the data, resulting in the formation of more clusters than intended.

3.4 Quantitative evaluation

The results of the quantitative analysis are presented in the below printed table:

```
    Clustering_Method         Adjusted Rand Index    Normalised Mutual Information
---------------------------   -------------------    -----------------------------
{'K-means'                }         0.68357                     0.77210
{'Fuzzy C-means'          }         0.69157                     0.77522
{'Possibilistic C-means'  }         0.39240                     0.57000
{'Probabilistic'          }         0.78124                     0.83518
{'Hierarchical (complete)'}         0.58910                     0.71878
{'Hierarchical (WPGMC)'   }         0.43998                     0.65373
{'Hierarchical (ward)'    }         0.65453                     0.76373
```

The table shows the adjusted rand index (ARI) and normalised mutual information (NMI) for each of the clustering methods used. The ARI and NMI are commonly used metrics to evaluate the performance of clustering algorithms, as they are able to adjust for chance and provide a measure of the similarity between the true labels and the predicted labels.

The k-means and fuzzy c-means algorithms achieved the second highest ARI and NMI scores among the cost function optimisation algorithms, with ARI of 0.68357 & 0.69157 and NMI of 0.77210 & 0.77522 respectively, while the probabilistic algorithm achieved the highest ARI and NMI scores, with ARI of 0.78124 and NMI of 0.83518. This suggests that these algorithms are able to produce clusters that are very similar to the true labels.

On the other hand, the Possibilistic c-means algorithm achieved lower scores with an ARI of 0.39240 and an NMI of 0.57000. This is to be expected, as the parameter tuning process was not as robust as for the other cost function optimisation algorithms.

The hierarchical algorithms, complete linkage, weighted pair group method with arithmetic mean (WPGMC) and ward linkage, achieved ARI scores of 0.58910, 0.43998 & 0.65453 and MRI scores of 0.71878, 0.65373 & 0.76373 respectively. The Ward algorithm clearly outperforms the others, while the WPGMC performs rather poorly. The latter is to be expected as there was an instance of non-monotonicity in the respective dendrogram (see Figure 10).

4 Concluding remarks

This study examined the high-resolution Salinas dataset. Given the challenges of clustering hyperspectral images due to their large size, dimensionality reduction to two principal components was applied, which retained more than 99% of the original data. However, the results of the bi-plot analysis [6] showed that the majority of the features could not be reconstructed after dimensionality reduction. This limitation is a necessary trade-off due to the inherent challenge of the curse of dimensionality.

A variety of clustering algorithms were employed, including both cost function optimisation and hierarchical approaches. The results of the quantitative analysis showed that the probabilistic algorithm, which uses a naive Bayesian approach, outperformed the other algorithms. Shortcomings of the other algorithms were also identified through qualitative analysis, with the most notable failures stemming from the presence of noise and lack of data compatibility.

References

[1] Pablo A Estévez et al. "Normalized mutual information feature selection". In: *IEEE Transactions on neural networks* 20.2 (2009), pp. 189–201. DOI: 10.1109/TNN.2008.2005601.

[2] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. "Clustering". In: *Mining of massive data sets*. Cambridge university press, 2020. Chap. 7, pp. 240–280. URL: http://mmds.org/#ver30.

[3] Douglas Steinley. "Properties of the hubert-arable adjusted rand index." In: *Psychological methods* 9.3 (2004), p. 386. DOI: https://doi.org/10.1037/1082-989X.9.3.386.

[4] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern recognition*. Elsevier, 2006.

[5] Scott I Vrieze. "Model selection and psychological theory: a discussion of the differences between the Akaike information criterion (AIC) and the Bayesian information criterion (BIC)." In: *Psychological methods* 17.2 (2012), p. 228. DOI: https://doi.org/10.1037/a0027127.

[6] Weikai Yan and Nicholas A Tinker. "Biplot analysis of multi-environment trial data: Principles and applications". In: *Canadian journal of plant science* 86.3 (2006), pp. 623–645. DOI: https://doi.org/10.4141/P05-169.

5 Appendix

Listing 1: Clustering analysis script

```
%% Clustering Algorithms, Project
%  Image Clustering on the Salinal valley data.
%
%  This script makes use of the following provided functions:
%
%     rand_data_init.m
%     k_means.m
%     possibi.m
%
%  Two further functions, written by the authors, were also used
%  for editting plot variables, namely:
%
%     PlotDimensions.m
%     ChangeInterpreter.m
%
%  In addition, many other standard MATLAB functions were
%  also utilized throughout the script to aid in the clustering
%  process.

```

```
%% Importing and processing data
clear
format compact
close all

load Salinas_Data

% Size of the Salinas cube
[p, n, l] = size(Salinas_Image);

% Making a two dimensional array whose rows correspond to the
% pixels and the columns to the bands, containing only the pixels
% with nonzero label.
X_total = reshape(Salinas_Image, p*n, l);
L       = reshape(Salinas_Labels, p*n, 1);

% This contains 1 in the positions corresponding to pixels
% with known class label.
existed_L = (L > 0);
X         = X_total(existed_L, :);

% The class labels.
y = L(existed_L);

% Abbreviations px = no. of rows (pixels) and nx = no. of columns
% (bands).
[px, nx] = size(X);

% Normalize each column of the dataset to its mean
% and also normalise to it standard deviation.
mean_X = mean(X);
s      = std(X);
X      = X - mean_X ./ s;

% Find the optimal number of components to retain.
[coeff, score, latent, tsquared, explained] = pca(X);
explained_variance = 0;
k = 0;

while explained_variance < 99
    k = k + 1;
    explained_variance = explained_variance + explained(k);
end


% Print original datase alongside the recundstructed dataset
% in order to assess reconstructibility (explained variance
% does not necessarilly imply reconstructibility).
figure; biplot(coeff(:,1:k), 'scores', score(:,1:k));


% Reduce the dataset to the optimal number of components
```

```
X = score(:, 1:k);

% Find the optimal number of clusters using the Calinski-Harabasz
% criterion and the 'elbow' method.
myfunc = @(X,K)(kmeans(X, K, 'Distance', 'cityblock',...
                           'MaxIter', 1000));
evaluation = evalclusters(X, myfunc, "CalinskiHarabasz",...
                        "KList", 1:20);
figure; plot(evaluation);

% Initialize the vector to store the within-cluster
% sum of squares (WCSS).
wcss = zeros(1, 20);

options = statset('UseParallel', 1, 'MaxIter', 1000);
for i = 1:20
    % Perform k-means clustering
    [~, ~, sumd] = kmeans(X, i, 'Distance', 'cityblock', 'Options',
       options);

    % Store the WCSS for each value of k
    wcss(i) = sum(sumd);
end

% Plot the WCSS against the number of clusters.
figure;
axes1 = axes('Parent', gcf);
hold(axes1, 'on');

% Create line.
line(1:20, wcss, 'Parent', axes1, 'MarkerSize', 10, 'Marker', '.');

% The 'elbow' point is the point at which the WCSS starts to plateau
% as the optimal number of clusters, as it indicates that adding
   more
% clusters will not significantly improve the WCSS.

% Calculate the second derivative of the WCSS
d2 = diff(diff(wcss));

% Find the local maxima of the second derivative
[~, idx] = findpeaks(-d2);

% The optimal number of clusters is the number of clusters where the
    second
% derivative changes sign
k1 = idx(1) + 2;
plot(k1, wcss(k1), 'Marker', 'o', 'Color', [0 0 1]);

xlabel('Number of Clusters');
ylabel('Within-Cluster Sum of Squares');
box(axes1,'on');
```

```
hold(axes1,'off');

% Prompt optimal number of clusters as calculated by using the
% Calinski-Harabasz criterion and the Elbow method.
prompt = sprintf('Enter the number of clusters k (Calinski-Harabasz
   criterion = %d, Elbow Method = %d): ', evaluation.OptimalK, k1);
k = input(prompt);

%% Cost Function Optimisation Algorithms

% K-means
options     = statset('UseParallel', 1, 'MaxIter', 1000);
num_repeats = 100;
min_sumd    = inf;
for i = 1:num_repeats
    [idx,C,sumd,D] = kmeans(X, k, 'Start', 'sample', 'Options',
       options);
    if min_sumd > sum(sumd)
        min_sumd = sum(sumd);
        cl_label = idx;
    end
end

cl_label_tot            = zeros(p*n, 1);
cl_label_tot(existed_L)= cl_label;
im_cl_label             = reshape(cl_label_tot, p, n);

% Fuzzy C-means
num_repeats = 2;
min_cost    = inf;
options     = [2; 1000; 1e-5; 1];
fuzzifier_values = 1.1:0.1:2;

for i = 1:numel(fuzzifier_values)
    options(1) = fuzzifier_values(i);
    for j = 1:num_repeats
        [center, U, obj_fcn] = fcm(X, k, options);
        if min_cost > min(obj_fcn)
            min_cost = obj_fcn;
            [~, cl_label1] = max(U, [], 1);
            U1 = U;
        end
    end
end

cl_label_tot             = zeros(p*n,1);
cl_label_tot(existed_L)  = cl_label1;
im_cl_label2             = reshape(cl_label_tot,p,n);

% Possibilistic C-means
eta = ones(1,k);  % Eta parameters of the clusters
q = 40;           % q parameter of the algorithm
```

```matlab
sed = 1;           % Seed for random generator
init_proc = 2;     % Use "rand_data_init" initialization procedure
e_thres = 0.0001; % Threshold for termination condition

% Run the possibilistic clustering algorithm
[U, theta] = possibi(X', k, eta, q, sed, init_proc, e_thres);

% Assign each data point to the cluster with the highest
   compatibility
[~, cl_label2] = max(U, [], 2);

cl_label_tot(existed_L)=cl_label2;
im_cl_label3=reshape(cl_label_tot,p,n);


% Probabilistic
ks        = 2:1:12;
options   = statset('MaxIter', 1000);
best_BIC = inf;
for i = 1:length(ks)
    k = ks(i);
    options.Start = 'kmeans';
    gm   = fitgmdist(X, k, 'CovarianceType', 'full', 'Options',
       options);
    BIC = gm.BIC;
    if BIC < best_BIC
        best_BIC    = BIC;
        best_model = gm;
    end

    options.Start = 'kmeans';
    gm = fitgmdist(X,k,'CovarianceType','diagonal','Options',options
       );
    BIC = gm.BIC;
    if BIC < best_BIC
        best_BIC    = BIC;
        best_model = gm;
    end
end

cl_label3                = cluster(best_model, X);
cl_label_tot             = zeros(p*n, 1);
cl_label_tot(existed_L) = cl_label3;
im_cl_label4             = reshape(cl_label_tot, p, n);

%% Hierarchical Algorithms

% Metric pool from which hierachical algorithms will choose from.
metrics = {'euclidean', 'cityblock', 'mahalanobis', 'chebychev'};

% Complete linkage.
max_coph_corr = -inf;
```

```
best_metric = '';
best_Z = [];

for i = 1:length(metrics)
    Y = pdist(X, metrics{i});
    Z = linkage(Y, 'complete');
    coph_corr = cophenet(Z, Y);
    if coph_corr > max_coph_corr
        max_coph_corr = coph_corr;
        best_metric   = metrics{i};
        best_Z = Z;
    end
end

fprintf('The best metric, for complete linkage, is %s with a
   cophenetic correlation coefficient of %f\n', best_metric,
   max_coph_corr);
figure;

cl_label4                  = cluster(best_Z, 'maxclust', k);
cl_label_tot               = zeros(p*n, 1);
cl_label_tot(existed_L) = cl_label4;
im_cl_label5               = reshape(cl_label_tot, p, n);

% Print dendogram.
figure;
dendogram(Z, 0);
set(gca, 'xticklabel', []);


% WPGMC linkage.
Y  = pdist(X, metrics{1});
Z1 = linkage(Y, 'WPGMC');

cl_label5                  = cluster(Z1, 'maxclust', k);
cl_label_tot               = zeros(p*n, 1);
cl_label_tot(existed_L) = cl_label5;
im_cl_label6               = reshape(cl_label_tot, p, n);

% Print dendogram.
figure;
dendogram(Z1, 0);
set(gca, 'xticklabel', []);


% Ward linkage.
Y  = pdist(X, metrics{1});
Z2 = linkage(Y, 'ward');

cl_label6                  = cluster(Z2, 'maxclust', k);
cl_label_tot               = zeros(p*n, 1);
cl_label_tot(existed_L) = cl_label6;
```

```
im_cl_label7            = reshape(cl_label_tot, p, n);

% Print dendogram.
figure;
dendogram(Z2, 0);
set(gca, 'xticklabel', []);

%% Qualitative Evaluation

% Plot the reconstructed images for qualitative evaluation.
figure;

subplot(4,2,1);
imagesc(Salinas_Labels); axis off;
title("Original representation");

subplot(4,2,2);
imagesc(im_cl_label); axis off;
title('K-means');

subplot(4,2,3);
imagesc(im_cl_label2); axis off;
title('Fuzzy C-means');

subplot(4,2,4);
imagesc(im_cl_label3); axis off;
title('Possibilistic C-means');

subplot(4,2,5);
imagesc(im_cl_label4); axis off;
title('Probabilistic');

subplot(4,2,6);
imagesc(im_cl_label5); axis off;
title('Complete-link');

subplot(4,2,7);
imagesc(im_cl_label6); axis off;
title('WPGMC');

subplot(4,2,8);
imagesc(im_cl_label7); axis off;
title('Ward');

% Perform PCA.
[coeff, score] = pca(X);

% Plot the first two principal components colored by the original
    labels.
figure;
scatter(score(:,1), score(:,2), [], y, 'filled', 'MarkerEdgeColor',
    'k');
```

```
title('PCA result with Original Figure');
xlabel('First Principal Component');
ylabel('Second Principal Component');
colorbar()

% Plot the first two principal components colored by the cluster
    labels
% obtained by K-means.
figure;
scatter(score(:,1), score(:,2), [], cl_label, 'filled', '
    MarkerEdgeColor', 'k');
title('PCA result with K-means Clustering');
xlabel('First Principal Component');
ylabel('Second Principal Component');
colorbar()

% Plot the first two principal components colored by the cluster
    labels
% obtained by Fuzzy C-means.
figure;
scatter(score(:,1), score(:,2), [], cl_label1', 'filled', '
    MarkerEdgeColor', 'k');
title('PCA result with Fuzzy C-means Clustering');
xlabel('First Principal Component');
ylabel('Second Principal Component');
colorbar()

% Plot the first two principal components colored by the cluster
    labels
% obtained by Possibilistic C-means.
figure;
scatter(score(:,1), score(:,2), [], cl_label2, 'filled', '
    MarkerEdgeColor', 'k');
title('PCA result with Possibilistic C-means Clustering');
xlabel('First Principal Component');
ylabel('Second Principal Component');
colorbar()

% Plot the first two principal components colored by the cluster
    labels
% obtained by Probabilistic Clustering.
figure;
scatter(score(:,1), score(:,2), [], cl_label3, 'filled', '
    MarkerEdgeColor', 'k');
title('PCA result with Probabilistic Clustering');
xlabel('First Principal Component');
ylabel('Second Principal Component');
colorbar()

% Plot the first two principal components colored by the cluster
    labels
% obtained by Hierarchical Clustering (complete linkage).
```

```matlab
figure;
scatter(score(:,1), score(:,2), [], cl_label4, 'filled', '
    MarkerEdgeColor', 'k');
title('PCA result with Hierarchical Clustering (complete linkage)');
xlabel('First Principal Component');
ylabel('Second Principal Component');
colorbar()

% Plot the first two principal components colored by the cluster
    labels
% obtained by Hierarchical Clustering (WPGMC linkage).
figure;
scatter(score(:,1), score(:,2), [], cl_label5, 'filled', '
    MarkerEdgeColor', 'k');
title('PCA result with Hierarchical Clustering (WPGMC linkage)');
xlabel('First Principal Component');
ylabel('Second Principal Component');
colorbar()

% Plot the first two principal components colored by the cluster
    labels
% obtained by Hierarchical Clustering (ward linkage).
figure;
scatter(score(:,1), score(:,2), [], cl_label6, 'filled', '
    MarkerEdgeColor', 'k');
title('PCA result with Hierarchical Clustering (ward linkage)');
xlabel('First Principal Component');
ylabel('Second Principal Component');
colorbar()

% Configure and save all figures at the end by using a loop to
    iterate
% over all figure objects.
for i = 1:numel(findobj('type', 'figure'))
    PlotDimensions(figure(i), 'centimeters', [15.747, 16], 12);
    ChangeInterpreter(figure(i), 'LaTeX');
    exportgraphics(gcf, sprintf('plot%d.pdf', i), 'BackgroundColor',
        'none');
end

% Gather all the clustering labels.
cl_labels = {cl_label, cl_label1,...
             cl_label2, cl_label3,...
             cl_label4, cl_label5,...
             cl_label6};

%% Quantitative Evaluation

% Initialise evaluation indices.
ARI = zeros(1, length(cl_labels));
NMI = zeros(1, length(cl_labels));
for i = 1:length(cl_labels)
```

```
    % Adjusted Rand Index (ARI).
    ARI(1, i) = rand_index(y, cl_labels{i}, 'adjusted');

    % Normalised Mutual Information (NMI).
    NMI(1, i) = nmi(y, cl_labels{i});
end

% Print the quantitative evaluation results in a table T.
clustering_methods = {'K-means', 'Fuzzy C-means',...
                      'Possibilistic C-means', 'Probabilistic',...
                      'Hierarchical (complete)',...
                      'Hierarchical (WPGMC)', 'Hierarchical (ward)'
                         };

T = table(clustering_methods', ARI', NMI',...
          'VariableNames', {'Clustering_Method',...
          'Adjusted Rand Index', 'Normalised Mutual Information'});
```

Listing 2: MATLAB® function for changing the interpreter of all objects within a figure.

```
function ChangeInterpreter(h, Interpreter)
% ChangeInterpreter() changes the interpreter of figure h.

    % Find all string type objects
    TexObj = findall(h, 'Type', 'Text');
    LegObj = findall(h, 'Type', 'Legend');
    AxeObj = findall(h, 'Type', 'Axes');
    ColObj = findall(h, 'Type', 'Colorbar');

    Obj = [TexObj; LegObj]; % Tex and Legend opbjects can be treated
        similarly
    n_Obj = length(Obj);
    for i = 1:n_Obj
        Obj(i).Interpreter = Interpreter;
    end

    Obj = [AxeObj; ColObj]; % Axes and ColorBar objects can be
       treated similarly
    n_Obj = length(Obj);
    for i = 1:n_Obj
        Obj(i).TickLabelInterpreter = Interpreter;
    end
end
```

Listing 3: MATLAB® function for configuring a figure's appearance options.

```
function PlotDimensions(h, Units, Plotsize, Fontsize)
% PlotDimensions() changes the string units, the fontsize
% and the unit size of the figure h.

    h.Units = Units; % measurement units
    h.Position(2) = (h.Position(2) - 8.5); % bottom-left corner of
```

```
        plot
    h.Position((3:4)) = Plotsize; % usually [15.747, 9]
    set(findall(h, '-property', 'FontSize'), 'FontSize', Fontsize);
        % fontsize
end
```