CS-E4650 Methods of Data MIning

Home assignment 1

Rosales Rodríguez, Pablo - 914769

Contents

Introduction

The assignment was performed using Python 3.7 in the Anaconda environment, exploiting Spyder as IDE software. The source code was attached with this report. The packages used on Python are the following ones:

```
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn import metrics
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import math
from scipy.spatial import distance
import statistics
import pandas as pd
```

Exercise 1

Initially, the data in csv format was loaded into the Python program and those columns with non-numerical information were removed. The cells containing NA values were filled using the mean of the column they are in. Finally, the resulting file was transformed into an array, easing its treatment in Python.

```
# The data is loaded from the csv file
df = pd.read_csv("nba2013_data.csv")

#Those columns containing non-numerical features are pruned
df = df.drop(columns = ['player', 'pos', 'bref_team_id', 'season'])
#Those values equal to NA are replaced with the mean of their column
df.fillna(df.mean(), inplace=True)
#The data is transformed into an array
X = np.array(df)
```

a)

k-means is one of the most popular clustering methods. It is based on solving an optimization problem in which the distance from each of the points to the representative or centroid of its group is the minimum. The k stands for the number of clusters in which the data must be grouped. As stated in the Data Mining Textbook, by Charu C. Aggarwal (Aggarwal, 2015), the optimization problem has the following expression:

$$Dist(\bar{X}_i, \bar{Y}_j) = ||\bar{X}_i, \bar{Y}_j||_2^2$$

In this exercise, k-means clustering method was carried out with different values of k using the following expression, in which k is the number of groups. The centroid of each cluster and the label for each of the data points are also extracted.

```
kmeans = KMeans(n_clusters=k, random_state=0).fit(X)
centroids = kmeans.cluster_centers_
labels = kmeans.labels_
```

After performing clusterization, three evaluation measurements were computed:

- Silhouette coefficient: Represents how high the "intra-cluster" similarity and "inter-cluster" are. It takes values inside [-1, 1]. As stated in (Aggarwal, 2015), the equation for computing the Silhouette

coefficient is the following:

$$S_i = \frac{Dmin_i^{out} - Davg_i^{in}}{max(Dmin_i^{out}, Davg_i^{in})}$$

Where, $Davg_i^{in}$ is the average of the distances measured from X_i to all the points within its cluster. Same computation is performed between X_i and the points in different clusters. Then, the minimum of those is selected as $Dmin_i^{out}$ (Aggarwal, 2015). The greater the result of the coefficient, the more robust the clusterization is, meaning that there is high "intra-cluster" similarity and low "inter-cluster" similarity.

- Calinski-Harabasz index *"evaluates the cluster validity based on the average between and within cluster sum of squares"* (Liu et al., 2010). The higher the value, the better is the clusterization.
- Davies-Bouldin index has a more complex expression than the previous ones. It first computes the similarity as the ratio of the average of the "intra-cluster" distances and the average of the "inter-clusters" distances. Then, the average of the most similar clusters is calculated (Scikit-learn developers, 2007-2020). The lower the score, the better the clusterization is.

The scores are computed as following:

```
#Goodness of clustering using different metrics
#Silhouette Coefficient
silhouette_result = metrics.silhouette_score(X, labels, metric='euclidean')
print("\nSilhouette Coefficient")
print(silhouette_result)
#Calinski Harabasz
print("\nCalinski-Harabasz Score:")
calinski_result = metrics.calinski_harabasz_score(X, labels)
print(calinski_result)
#Davies-Bouldin
print("\nDavies-Bouldin Score:")
bouldin_result = metrics.davies_bouldin_score(X, labels)
print(bouldin_result)
```

The results for each of the k values are the following: ### k = 10

- Silhouette Coefficient = 0.3518
- Calinski Harabasz score = 780.1749
- Davies-Bouldin score = 1.1533

k = 5

- Silhouette Coefficient = 0.4282
- Calinski Harabasz score = 1020.3352
- Davies-Bouldin score = 0.9442

k = 2

- Silhouette Coefficient = 0.5869
- Calinski Harabasz score = 1109.0588
- Davies-Bouldin score = 0.5922

In the given case, the results are better when the number of clusters is equal to two, since it gives a Silhouette Coefficient closer to one, a higher Calinski Harabasz Score and a lower Davies-Bouldin Score than the ones obtained in the other situations. Performing the same analysis, the clusterization result when the number of groups is equal to five is better than the case with k=10.

b)

In the following figure, there is a representation of the three scores considering a k contained between 2 and 20, with steps of 1 units. Given the Silhouette coefficient, since the closer to 1 is the score, the better, the desired value for k should be the lowest allowed. However, considering the initial tendency of the graph, k=7 (local maximum) can be considered an optimal number of clusters. For k=7, the coefficient would be 0.4199. Looking at the Davies-Bouldin score, the maximum value would be the optimal one. Again, considering the tendency, the maximum would be close to 18 groups, that gives a result of 1.1988. As a conclusion, the optimal number of clusters depends on the coefficient that is used for measuring the performance. Considering a compromise between the suggested three scores, a good number of clusters seems to be 7, for which the Silhouette Score and the Davies-Bouldin one reach a local maximum and there is an inflection point in the Calinski Harabasz curve.

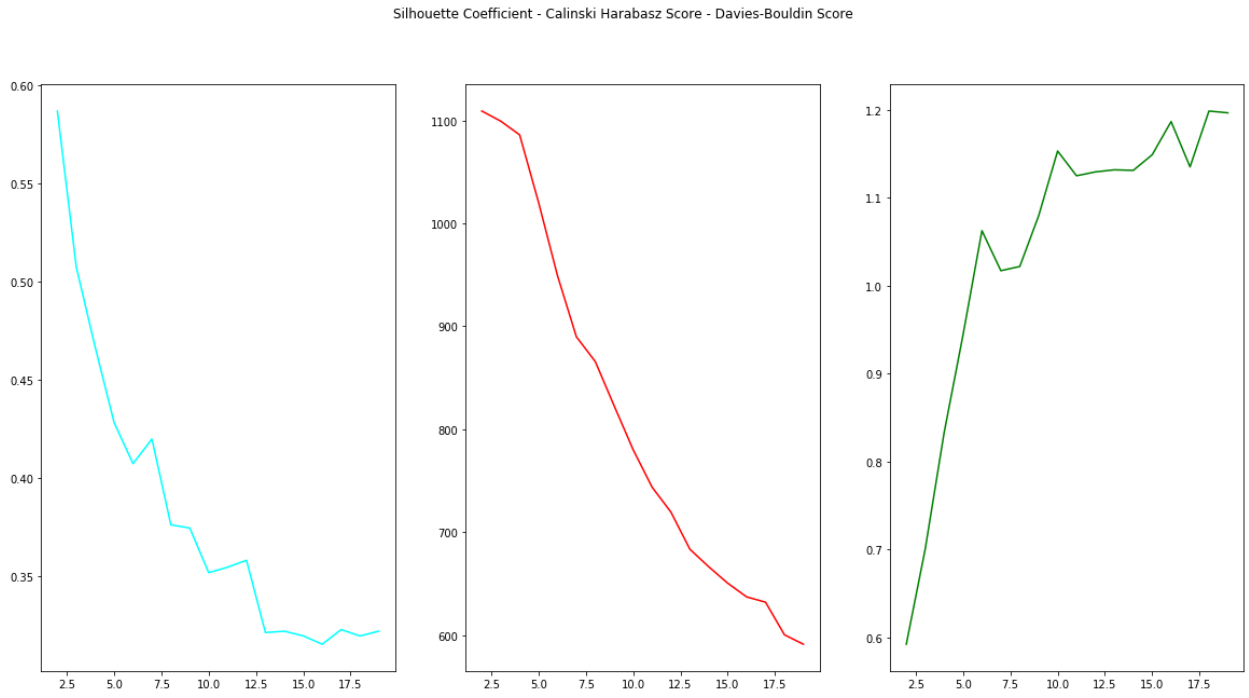

Figure 1: Scores for k between 2 and 20.

Exercise 2

Using the same data than that of exercise 1, hierarchical agglomerative clustering is performed with different linkage metrics. The agglomerative approach is characterized because the clustering starts with the individual points, grouping them in a *bottom-up* strategy, so in each iteration, clusters are merged with the ones from the higher hierarchical level. Four different strategies are implemented in this exercise. As stated in (The SciPy Community, 2020):

- Single-linkage metric: Computes the distance as the minimum between the points in cluster u and those in cluster v:

  $$d(u, v) = min(dist(u[i], v[j]))$$

  Where i represents all points in cluster u and j represents all points in cluster v.

- Complete-linkage metric: Analogous to the previous one, but using the maximum distance:

  $$d(u, v) = max(dist(u[i], v[j]))$$

Where i represents all points in cluster u and j represents all points in cluster v.

- Average-linkage metric: computes the average as following:

$$d(u, v) = \sum_{ij} \frac{d(u[i], v[j])}{(|u| * |j|)}$$

- Distance of centroids metric: considering c_s and c_t the centroids of the clusters s and t:

$$dist(s, t) = ||c_s - c_t||_2$$

Making use of the previously explained metrics, the clusterization was performed as following:

```
#Hierarchical agglomerative clustering using sklearn library
#Single-linkage metric
clustering_single = AgglomerativeClustering(n_clusters=k, linkage="single").fit(X)
single_labels = clustering_single.labels_
#Complete-linkage metric
clustering_complete = AgglomerativeClustering(n_clusters=k, linkage="complete").fit(X)
complete_labels = clustering_complete.labels_
#Average-linkage metric
clustering_average = AgglomerativeClustering(n_clusters=k, linkage="average").fit(X)
average_labels = clustering_average.labels_
#Distance of centroids metric
#Array containing an upper triangular of the distance matrix
Y = sp.spatial.distance.pdist(X, metric = "euclidean")
#Performs the clustering with the "distance of centroids" metric
Z = sp.cluster.hierarchy.linkage(Y, method = "centroid")
#Transformation in flat clusters from the previous hierarchical clustering
Z = sp.cluster.hierarchy.fcluster(Z, k, criterion = "maxclust")
```

Then, in order to find out the optimal metric for this data, the silhouette coefficient was computed for several values of the number of clusters (k), giving as result the following figure, in which the obtained curve for each of the metrics (single-linkage, complete-linkage, average-linkage and distance of centroids) is showed.

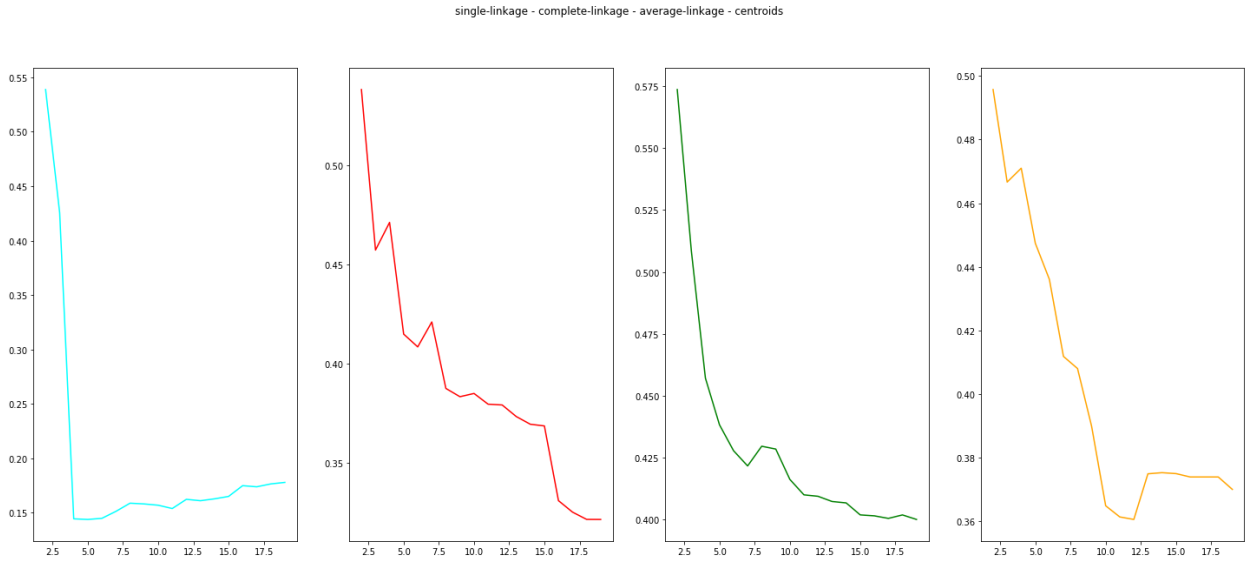

Figure 2: Silhouette coefficient for each of the metrics with k between 2 and 20.

Considering the previous figure, the optimal metric is that providing a higher result of the silhouette coefficient. In the given case, the complete-linkage metric is the one for which, in general, the value of the score is higher, offering several local maxima for which the number of clusters could be considered as optimal. However, for all of the previously explained metrics, the coefficient drops when the number of clusters is over 17.

Exercise 3

After shuffling the data and repeating the previous exercise, the goodness measures did not change. However, there should be some changes because hierarchical clustering algorithms (mainly single-linkage and complete-linkage) begin by computing the minimum distance, and when having to equal minimum distances, there must be some stochasticity when the decision of which one to choose in order to enable the initial clustering action has to be made.

Exercise 4

To carry out exercise 4, a csv file was created containing the information provided on the table.

a)

Initially, the data from the csv was loaded on the Python script. In order to compute the pairwise distances, the information was normalized between 0 and 1. The obtained distance matrix is the following:

	0	1	2	3	4	5
0	0	0.285714	0.520008	0.362656	0.937079	1.2289
1	0.285714	0	0.520008	0.542941	1.09756	1.41421
2	0.520008	0.520008	0	0.330772	0.661545	0.992317
3	0.362656	0.542941	0.330772	0	0.575876	0.878052
4	0.937079	1.09756	0.661545	0.575876	0	0.330772
5	1.2289	1.41421	0.992317	0.878052	0.330772	0

Figure 3: Euclidean Matrix.

As expected, the values in the diagonal are zero and the matrix is symmetric. As depicted on the figure, the maximum distance is that between cow 1, Sunny and cow 5, Molly. It is due to the fact that Sunny is one of the two youngest cows and Molly is the oldest one, as well as to the fact that Sunny gives only 10 units of milk per day, while Molly provides 45. The distances between other cows are lower than the one between the mentioned two.

b)

The numerical features were removed from the table and the Goodall Measure was computed to get the pairwise similarities. As stated in the book (Aggarwal, 2015), the expression for computing the similarity following the Goodall measure is:

$$S(x_i, y_i) = \begin{cases} 1 - p_k(x_i)^2 & if \quad x_i = y_i \\ 0 & otherwise \end{cases}$$

Where p_k is *"the fraction or records in which the kth attribute takes on the value of x in the data set"* (Aggarwal, 2015). The obtained matrix is the following:

	0	1	2	3	4	5
0	0.916667	0.296296	0.296296	0	0	0
1	0.296296	0.842593	0	0.25	0	0.546296
2	0.296296	0	0.842593	0.25	0.25	0.296296
3	0	0.25	0.25	0.796296	0.546296	0.25
4	0	0	0.25	0.546296	0.87037	0
5	0	0.546296	0.296296	0.25	0	0.842593

Figure 4: Goodall Matrix.

Considering the information provided on the table, the results obtained in the Goodall Matrix make sense, since the most similar cows, when considering categorical data, are Daisy and Strawberry: They have the same character and music taste. This is why they have the highest value inside the matrix (without considering the diagonal ones, which compare each cow with itself). Comparing Holstein with Daisy, Strawberry or Molly, the similarity is null, since there are no coincidences at all.

c)

The goal of 4c is creating a similarity measure combining the numerical and categorical data. In the previous questions, each of them was treated independently. For the numerical data, the matrix of euclidean distances was computed. For the categorical data, the matrix of Goodall measure was computed. In order to combine the information from both, the numerical data must be expressed in terms of a similarity, instead of distance. This is the reason why the cosine similarity matrix was computed. However, if the euclidean matrix is computed with normalized vectors (in terms of their norm, unit vectors), then, the cosine of the two vectors and the distance are related as following:

$$Dist_{euclidean}(\boldsymbol{x}, \boldsymbol{y}) = 2(1 - cos(\boldsymbol{x}, \boldsymbol{y}))$$

The cosine matrix is the following:

	0	1	2	3	4	5
0	1	0.995229	0.975441	0.998261	0.995229	0.997093
1	0.995229	1	0.992278	0.99925	1	0.99977
2	0.975441	0.992278	1	0.986729	0.992278	0.989388
3	0.998261	0.99925	0.986729	1	0.99925	0.999851
4	0.995229	1	0.992278	0.99925	1	0.99977
5	0.997093	0.99977	0.989388	0.999851	0.99977	1

Figure 5: Cosine Matrix.

The way similarity is computed in mixed data is:

$$Sim_{mixed}(\boldsymbol{x}, \boldsymbol{y}) = \lambda \frac{Sim_{numerical}}{\sigma_N} + (1 - \lambda) \frac{Sim_{categorical}}{\sigma_C}$$

Where σ_N and σ_C are the standard deviations of the numerical and categorical data respectively. The similarity matrix was computed as following:

```
#Mixed data similarity measure
mixed_similarity_matrix = np.zeros((len(df_cows_num_array), len(df_cows_num_array)))
for i in range(len(df_cows_num_array)):
    for j in range(len(df_cows_num_array)):
        mixed_similarity_matrix[i][j] = (lambda_var)*cosine_matrix[i][j]/stdev_num...
        ...+ (1-lambda_var)*overlap_matrix[i][j]/stdev_cat
```

The result of the previous computation is showed in figure 6.

	0	1	2	3	4	5
0	64.3954	62.9675	61.7268	62.5932	62.4031	62.52
1	62.9675	64.3954	62.2181	63.2196	62.7023	63.8166
2	61.7268	62.2181	64.3954	62.4346	62.7825	62.6013
3	62.5932	63.2196	62.4346	64.3954	63.784	63.2573
4	62.4031	62.7023	62.7825	63.784	64.3954	62.6878
5	62.52	63.8166	62.6013	63.2573	62.6878	64.3954

Figure 6: Mixed data similarity matrix.

The matrix in figure 6 is a summary of the conclusions extracted from the previous questions, now, considering both, numerical and categorical data.

A different approach to this exercise consists in computing the similarity matrix linked to the euclidean distance one, by making use of the following expression:

$$s_d = 1 - d$$

Where s_d represents the similarity and d represents the distance. Hence, given the euclidean matrix from figure 3, the similarity matrix would be the following one:

	0	1	2	3	4	5
0	64.3954	45.3517	30.661	39.9629	3.94526	-14.3528
1	45.3517	64.3954	30.0966	29.223	-6.11754	-24.8433
2	30.661	30.0966	64.3954	42.5265	21.7863	1.04611
3	39.9629	29.223	42.5265	64.3954	27.7224	8.21082
4	3.94526	-6.11754	21.7863	27.7224	64.3954	41.9621
5	-14.3528	-24.8433	1.04611	8.21082	41.9621	64.3954

Figure 7: Mixed data similarity matrix with different approach.

As depicted in the previous figure, the most different cows are Sunny and Molly, since they have the lowest value in the lastly computed similarity matrix (cells (1,5) and (5,1)). It is reasonable, since, even though they have the character in common, the rest of the categorical data is different and the units of milk per day that Sunny gives are much lower than those coming from Molly, as well as the age.

d)

The previous similarity measure can be expressed in terms of distance using the equation:

$$s_d = \frac{1}{1+d}$$

So,

$$d = \frac{1 - s_d}{s_d}$$

The distance matrix for the mixed data is the one in figure 8.

	0	1	2	3	4	5
0	-0.984471	-0.97795	-0.967385	-0.974977	-0.746531	-1.06967
1	-0.97795	-0.984471	-0.966774	-0.96578	-1.16346	-1.04025
2	-0.967385	-0.966774	-0.984471	-0.976485	-0.9541	-0.0440754
3	-0.974977	-0.96578	-0.976485	-0.984471	-0.963928	-0.878209
4	-0.746531	-1.16346	-0.9541	-0.963928	-0.984471	-0.976169
5	-1.06967	-1.04025	-0.0440754	-0.878209	-0.976169	-0.984471

Figure 8: Mixed data distance matrix.

It is not a metric, since it does not satisfies all the requirements. Even though it is symmetric, it happens that:

$$d(\boldsymbol{x}, \boldsymbol{y}) \ngeq 0$$

And given the equation for computing the distance, could happen that:

$$d(\boldsymbol{x}, \boldsymbol{y}) = 0 \quad when \quad \boldsymbol{x} \neq \boldsymbol{y}$$

Exercise 5

Considering the following data set:

x	y
0	1
-1/2	3/2
3/2	5/2
1	3

a)

In order to perform the PCA of the data set, first, the data is adjusted with respect to the average, so, given the sample means:

$$\bar{x} = \frac{1}{4}(0 - \frac{1}{2} + \frac{3}{2} + 1) = 0.5$$

$$\bar{y} = \frac{1}{4}(1 + \frac{3}{2} + \frac{5}{2} + 3) = 2$$

The adjusted data is:

x-$\bar{x}$	y-$\bar{y}$
-0.5	-1
-1	-0.5
1	0.5
0.5	1

The covariance matrix must be computed for data contained in the previous table. This is the expression for the covariance:

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

So, computing the covariance for each combination:

$$cov(X,X) = \frac{0.25+1+1+0.25}{3} = \frac{5}{6}$$

$$cov(X,Y) = cov(Y,X) = \frac{0.5+0.5+0.5+0.5}{3} = \frac{2}{3}$$

$$cov(Y,Y) = \frac{1+0.25+0.25+1}{3} = \frac{5}{6}$$

The covariance matrix is:

$$\begin{pmatrix} cov(X,X) & cov(X,Y) \\ cov(Y,X) & cov(Y,Y) \end{pmatrix} = \begin{pmatrix} \frac{5}{6} & \frac{2}{3} \\ \frac{2}{3} & \frac{5}{6} \end{pmatrix}$$

Eigenvalues and Eigenvectors of the covariance matrix:

$$det(\lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} \frac{5}{6} & \frac{2}{3} \\ \frac{2}{3} & \frac{5}{6} \end{pmatrix}) = \begin{vmatrix} \lambda - \frac{5}{6} & -\frac{2}{3} \\ -\frac{2}{3} & \lambda - \frac{5}{6} \end{vmatrix} = \lambda^2 - \frac{5}{3}\lambda + \frac{1}{4} = 0 \rightarrow \begin{cases} \lambda = \frac{3}{2} \\ \lambda = \frac{1}{6} \end{cases}$$

The Eigenvalues of the covariance matrix are $\lambda_1 = \frac{3}{2}$ and $\lambda_2 = \frac{1}{6}$. Given those eigenvalues, the eigenvectors can be computed as following:

$$v(\lambda_1) = v(\frac{3}{2}) = ((x,y)/(\lambda I - A)(x,y)^T = 0, \quad where \quad A \quad is \quad the \quad covariance \quad matrix)$$

$$\begin{pmatrix} \frac{2}{3} & -\frac{2}{3} \\ -\frac{2}{3} & \frac{2}{3} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow \begin{cases} \frac{2}{3}x - \frac{2}{3}y = 0 \\ -\frac{2}{3}x + \frac{2}{3}y = 0 \end{cases} \rightarrow x = y \rightarrow v(\lambda_1 = \frac{3}{2}) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$v(\lambda_2) = v(\frac{1}{6}) = ((x,y)/(\lambda I - A)(x,y)^T = 0, \quad where \quad A \quad is \quad the \quad covariance \quad matrix)$$

$$\begin{pmatrix} -\frac{2}{3} & -\frac{2}{3} \\ -\frac{2}{3} & -\frac{2}{3} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow \begin{cases} -\frac{2}{3}x - \frac{2}{3}y = 0 \\ -\frac{2}{3}x - \frac{2}{3}y = 0 \end{cases} \rightarrow x = -y \rightarrow v(\lambda_2 = \frac{1}{6}) = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

So far, the eigenvalues and their corresponding eigenvectors have been computed:

$$v_1 = v(\lambda_1 = \frac{3}{2}) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$v_2 = v(\lambda_2 = \frac{1}{6}) = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

b)

PCA has as main target performing dimensionality reduction. Initially, this is done using the eigenvector corresponding to the greatest eigenvalue:

$$Data_{final} = Eigenvector^T . Data_{adjusted}$$

In the given case, the greatest eigenvalue is $\lambda_1 = \frac{3}{2}$ and its corresponding eigenvector is:

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

So, the data set obtained after the transformation is:

$$z_1 = (1,1)(-0.5,-1) = -1.5$$

$$z_2 = (1,1)(-1,-0.5) = -1.5$$

$$z_3 = (1,1)(1,0.5) = 1.5$$

$$z_4 = (1,1)(0.5,1) = 1.5$$

z
-1.5
-1.5
1.5
1.5

Finally, transforming the original data set into a 2-dimensional representation:

$$z_1 = (1,1)(-0.5,-1) = -1.5$$

$$z_2 = (1,1)(-1,-0.5) = -1.5$$

$$z_3 = (1,1)(1,0.5) = 1.5$$

$$z_4 = (1,1)(0.5,1) = 1.5$$

$$q_1 = (-1,1)(-0.5,-1) = -0.5$$

$$q_2 = (-1,1)(-1,-0.5) = 0.5$$

$$q_3 = (-1,1)(1,0.5) = -0.5$$

$$q_4 = (-1,1)(0.5,1) = 0.5$$

z	q
-1.5	-0.5
-1.5	0.5
1.5	-0.5
1.5	0.5

c)

Euclidean distances computation:

$$d(x, y) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$$

For the original data set:

$$d_{12} = d_{21} = \frac{\sqrt{2}}{2} = 0.7071$$

$$d_{13} = d_{31} = 3\frac{\sqrt{2}}{2} = 2.1213$$

$$d_{14} = d_{41} = \sqrt{5} = 2.2361$$

$$d_{23} = d_{32} = \sqrt{5} = 2.2361$$

$$d_{24} = d_{42} = 3\frac{\sqrt{2}}{2} = 2.1213$$

$$d_{34} = d_{43} = \frac{\sqrt{2}}{2} = 0.7071$$

For the 1-dimensional representation

$$d_{12} = d_{21} = 0$$

$$d_{13} = d_{31} = 3$$

$$d_{14} = d_{41} = 3$$

$$d_{23} = d_{32} = 3$$

$$d_{24} = d_{42} = 3$$

$$d_{34} = d_{43} = 0$$

For the 2-dimensional representation

$$d_{12} = d_{21} = 1$$

$$d_{13} = d_{31} = 3$$

$$d_{14} = d_{41} = \sqrt{10} = 3.1623$$

$$d_{23} = d_{32} = \sqrt{10} = 3.1623$$

$$d_{24} = d_{42} = 3$$

$$d_{34} = d_{43} = 1$$

When removing one of the dimensions, remaining only the most relevant component (principal component), the points are placed in one line, so some of them overlap with each other, giving as a result that some of the distances are zero. Also, as a consequence of this transformation there is a symmetry with respect to the vertical axis.

When performing the 2-dimensional transformation, the symmetry happens to be with respect both axes, giving as a result a rectangle with a point in each of its corners. This is the reason why there are only three measures for the distances and the largest one is equal to the square root of the sum of the squares of the others.

d)

This is the new data set:

$$\begin{pmatrix} \sqrt{0.5} & \sqrt(0.5) \\ \sqrt{0.5} & 2\sqrt(0.5) \\ 4\sqrt(0.5) & \sqrt(0.5) \\ 4\sqrt(0.5) & 2\sqrt(0.5) \end{pmatrix}$$

Computing the average for each variable:

$$\bar{x} = 5\frac{\sqrt{2}}{4} = 1.7678$$

$$\bar{y} = 3\frac{\sqrt{2}4}{=}1.0607$$

The adjusted data set is:

$$\begin{pmatrix} -3\frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} \\ -3\frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} \\ 3\frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} \\ 3\frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} \end{pmatrix}$$

The covariance is computed as following

$$cov(X,X) = \frac{4x1.125}{3} = 1.5$$

$$cov(X,Y) = cov(Y,X) = \frac{0.375 - 0.375 + 0.375 - 0.375}{3} = 0$$

$$cov(Y,Y) = \frac{4x0.125}{3} = \frac{1}{6} = 0.1667$$

The covariance matrix is:

$$\begin{pmatrix} cov(X,X) & cov(X,Y) \\ cov(Y,X) & cov(Y,Y) \end{pmatrix} = \begin{pmatrix} 1.5 & 0 \\ 0 & \frac{1}{6} \end{pmatrix}$$

Solving:

$$\begin{vmatrix} \lambda - 1.5 & 0 \\ 0 & \lambda - \frac{1}{6} \end{vmatrix} = 0 \rightarrow \begin{cases} \lambda = \frac{3}{2} \\ \lambda = \frac{1}{6} \end{cases}$$

The eigenvalues have been computed and they result the same as in the previous data set. Now the eigenvectors are computed:

$$v(\lambda_1) = v(\frac{3}{2}) = ((x,y)/(\lambda I - A)(x,y)^T = 0, \quad where \quad A \quad is \quad the \quad covariance \quad matrix)$$

$$\begin{pmatrix} 0 & 0 \\ 0 & \frac{4}{3} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow y = 0 \rightarrow v(\lambda_1 = \frac{3}{2}) = (1,0)$$

$$v(\lambda_2) = v(\frac{1}{6}) = ((x,y)/(\lambda I - A)(x,y)^T = 0, \quad where \quad A \quad is \quad the \quad covariance \quad matrix)$$

$$\begin{pmatrix} -\frac{4}{3} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow x = 0 \rightarrow v(\lambda_2 = \frac{1}{6}) = (0,1)$$

Both data sets have the same principal components (same eigenvalues), but different eigenvectors. The eigenvectors represent the direction in which the data set changes the most, as it can be seen in the figure. The fact that the two covariance matrices are similar matrices means that both data sets have the same variance along their eigenvectors. Actually, from figure 9 it can be deduced that the variation of the first data set along the direction of its first eigenvector (1,1) between the first and the fourth point is equal to the one of the second data set, whose direction would be (0,1).

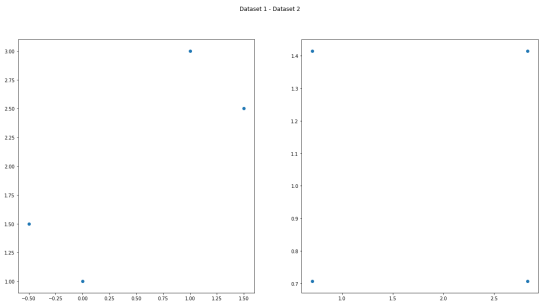

Figure 9: Data sets.

References

- Charu C. Aggarwal. (2015). Data Mining. The Textbook. Switzerland: Springer.
- Y. Liu, Z. Li, H. Xiong, X. Gao and J. Wu, "Understanding of Internal Clustering Validation Measures", 2010 IEEE International Conference on Data Mining, Sydney, NSW, 2010, pp.911-916, doi: 10.1109/ICDM.2010.35
- Scikit-learn developers (2007-2020). Sklearn.metrics.davies_bouldin_score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies_bouldin_score.html
- The SciPy Community (2020). Scipy.cluster.hierarchy.linkage. https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html