Lab 11. Unsupervised Learning

Presented by Andrés Carofilis
Prepared by Deisy Chaves and Enrique Alegre
Modified by Rocio Alaiz & by Enrique Alegre
Área de Ingeniería de Sistemas y Automática
Dpto. de Ing. Eléctrica y de Sistemas y Automática
Universidad de León (Spain)

Table of contents

0. Summary

In this Lab, we will train and test several unsupervised clustering methods, mainly using scikit-learn and SciPy functions. We will use the AirlinesCluster Dataset to find similar groups of customers belonging to an airline's frequent flyer program. The csv file containing the AirlinesCluster Dataset can be found in Agora.
First, we will understand and normalize the data before building the clusters. Later, we will test K-means and Hierarchical clustering methods and analyze the obtained results using *Pandas* library functions. Finaly, we will apply DBSCAN to see how many clusters this method proposes with different configurations.

What to prepare

Students will prepare a document with the questions that can be found in this handout and their corresponding answers. (1.3.1, 1.3.2, 1.4.1, 1.4.2, 2.1.1.1, 2.1.1.2, 2.1.2.1, 2.1.2.2, 2.2.1, 2.2.2, 2.2.3, 2.3.1, 2.3.2, 2.3.3). It is convenient to include all the plots related to the questions.
It is also convenient to create a Python script, divided into cells (using #%%) and properly commented.

With these two outcomes, the students will be prepared to answer the questions that will find in the Agora's questionnaire.

1. Problem definition, dataset description and data preparation

(Modified from https://en.wikipedia.org/wiki/Market_segmentation, https://github.com/bobbruno/Analytics-Edge/blob/master/Clustering%20Assignment/AirlinesCluster.md and others)

1.1. Problem definition

Market segmentation is the activity of dividing a business market, normally consisting of existing and potential customers, into groups or segments of consumers based on some type of similar characteristics. Grouping similar consumers allow marketers to target specific audiences in a cost-effective manner, reducing the risk of an unsuccessful or ineffective marketing campaign.

In this lab, we are going to find similar groups of customers who belong to an airline's frequent flyer program in order to help the airline to design their new trip offers. To achieve this goal, we will use machine learning clustering methods provided by scikit-learn library to fit a model that can identify groups (clusters) of customers given the information of the AirlinesCluster Dataset.

1.2. Dataset description

The AirlinesCluster Dataset contains information on 3,999 members of the frequent flyer program of an Airline. This dataset comes from the textbook "Data Mining for Business Intelligence," by Galit Shmueli, Nitin R. Patel, and Peter C. Bruce. It is formed by seven features or variables, described below:

1) **Balance:** number of miles eligible for award travel
2) **QualMiles:** number of *miles qualifying* for TopFlight status
3) **BonusMiles:** number of *miles earned* from non-flight bonus transactions in the past 12 months
4) **BonusTrans:** *number of* non-flight *bonus* transactions in the past 12 months
5) **FlightMiles:** number of *flight miles* in the past 12 months
6) **FlightTrans:** number of *flight transactions* in the past 12 months
7) **DaysSinceEnroll:** number *of days since enrolled* in the frequent flyer program

1.3. Dataset loading (pandas.read_csv)

(Adapted from https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html , https://www.dataquest.io/blog/sci-kit-learn-tutorial /)

The AirlinesCluster Dataset is available on the file AirlinesCluster.csv available in Agora. We will use the **pandas.read_csv()** function provided by the pandas module to read the csv file which contains comma separated values and convert that into a pandas DataFrame. A DataFrame is a tabular data-structure, where the first column contains the index which marks each row of data uniquely and the first row contains a label/name for each column, which are the original column names retained from the data set.

After loading the dataset, We will explore it by using the **pandas.DataFrame.head()** and **pandas.DataFrame.tail()** functions, that show the first and the last records of the imported dataset, respectively. Both methods have a parameter:

- **n:** Number of rows to select.

Understand the next code and solve the doubts that could come up.

Code:

```
#import necessary packages
import pandas as pd
import numpy as np

# load the AirlinesCluster dataset
url = "AirlinesCluster.csv"
airlines_dataset = pd.read_csv(url)

# suppress scientific float notation
np.set_printoptions(precision=5, suppress=True)

# visualization of the first few records of the dataset
print(airlines_dataset.head())
print(airlines_dataset.head(n=2))

# visualization of the last few records from the dataset
print(airlines_dataset.tail())
print(airlines_dataset.tail(n=2))

# display the different datatypes available in the dataset
print(airlines_dataset.dtypes)

# visualization of descriptive stats of the dataset
print(airlines_dataset.describe())
```

Questions:

- 1.3.1. Based on the previous result, which two features have (on average) the smallest values and the largest values?
- 1.3.2. Why is it important to normalize the data before clustering?

1.4. Data preprocessing ***(sklearn.preprocesing)***

(Adapted from https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing)

Clustering algorithms such as the K-means and Hierarchical, depend on calculating distances between different features. Thus, due to different scales of measurement of the variables some variables may have a higher influence on the clustering output. Therefore, standardizing the dataset is essential during clustering. For data standardization, we will use the **scale()** function from **sklearn.preprocesing** which has the next parameters:

- **X:** The data to center and scale.
- **axis:** The axis used to compute the means and standard deviations along. If 0, independently standardize each feature, otherwise (if 1) standardize each sample.
- **with_mean:** If True, center the data before scaling.
- **with_std:** If True, scale the data to unit variance (or equivalently, unit standard deviation).

Code:

```
# import the necessary packages
from sklearn import preprocessing

#standardize the data to normal distribution
dataset_standardized = preprocessing.scale(airlines_dataset)

# visualization of descriptive stats of the normalized dataset
dataset_standardized = pd.DataFrame(dataset_standardized)
print(dataset_standardized.describe())
```

Questions:

- 1.4.1. What are the mean and standard deviation of the features in the standardized dataset?
- 1.4.2. Based on the normalized dataset descriptive stats, which two features have (on average) the smallest values and the largest values?

2. Building and Using Unsupervised Clustering Models

scikit-learn is a library with a large number of implementations of supervised and unsupervised classifiers, making it easier to evaluate different algorithms and compare the performance of the models to see what works best for our data. During this Lab, we are going to evaluate different unsupervised clustering algorithms to build groups of customers from the *Airlines Cluster Dataset* which contains information of an airline's frequent flyer program (see section 1).

2.1. K-Means (sklearn.cluster.KMeans)

(Adapted from https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html)

KMeans implements the K-means algorithm for clustering data by trying to separate samples in K groups of equal variance, minimizing a stop criterion known as the inertia or within-cluster sum-of-squares. This algorithm requires the number of clusters to be specified. This class has the next parameters:

- **n_clusters**: The number of clusters to form as well as the number of centroids to generate.
- **init**: Method for centroids initialization. **'k-means++'** selects initial centroids in a smart way to speed up convergence. **'random'** choose k observations (rows) at random from data for the initial centroids.
- **n_init**: Number of times the k-means algorithm will be run with different centroid seeds. The final results will be the best output of n_init consecutive runs in terms of inertia.
- **max_iter**: Maximum number of iterations of the k-means algorithm for a single run.
- **tol**: Relative tolerance with regards to inertia to declare convergence.

2.1.1. Building and using the clustering model

The **KMeans** class from **sklearn.cluster.KMeans** module provided the functions **fit()** and **predict()** to build a new K-means clustering model and uses it to group new data**.**

The **fit (X)** function uses a parameter to compute a K-mean clustering:

- **X:** Training vectors to cluster.
- **sample_weight [optional]:** Weights applied to individual samples (1. for unweighted).

and return a K-mean clustering.

The **predict (X)** function uses the k-mean clustering fit previously to predicted the closest cluster each sample in X belongs to.

Study the next code, understand what it does and complete the missing lines. Try a little bit on your own before copying the provided solution on gray color.

Code:

```
#import necessary packages
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 7))

# visualization of the original data
plt.subplot(221)
plt.scatter(dataset_standardized[0], dataset_standardized[1])
plt.title("Original unclustered data")
```

```
# train the model using k=2
kmeans_model = KMeans(n_clusters=2, random_state=42)
kmeans_model.fit(dataset_standardized)
kmeans_predictions = kmeans_model.predict(dataset_standardized)

# visualization of the clustered data using k=2
plt.subplot(222)
plt.scatter(dataset_standardized[0], dataset_standardized[1], c=kmeans_predictions)
plt.title("Clustered data, k=2")

# train the model using k=3
kmeans_model = KMeans(n_clusters=3, random_state=42)
kmeans_model.fit(dataset_standardized)
kmeans_predictions = kmeans_model.predict(dataset_standardized)

# visualization of the clustered data using k=3
plt.subplot(223)
plt.scatter(dataset_standardized[0], dataset_standardized[1], c=kmeans_predictions)
plt.title("Clustered data, k=3")


# train the model using k=5
kmeans_model = KMeans(n_clusters=5, random_state=42)
kmeans_model.fit(dataset_standardized)
kmeans_predictions = kmeans_model.predict(dataset_standardized)

# visualization of the clustered data using k=5
plt.subplot(224)
plt.scatter(dataset_standardized[0], dataset_standardized[1], c=kmeans_predictions)
plt.title("Clustered data, k=5")
plt.show()


#visualization of descriptive stats of the clustered data using k=5
cluster_kmeans_data = pd.DataFrame(kmeans_predictions+1)
airlines_dataset['KmeansCluster'] = cluster_kmeans_data
kmeans_mean_cluster_pd =
pd.DataFrame(airlines_dataset.groupby('KmeansCluster').mean())
print(kmeans_mean_cluster_pd)
```

Questions
Considering the results obtained using k=5:

- 2.1.1.1 Compared to the other clusters, Cluster 1 has the largest average values in which features (if any)? Based on this, how would you describe the Airline's customers in Cluster 1?
- 2.1.1.2 Applied the previous analysis to the remaining four clusters and describe them.

2.1.2. Selecting the K parameter, The Elbow method

(Adapted from https://en.wikipedia.org/wiki/Elbow_method_(clustering), Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, and others)

The Elbow method was designed by Robert L. Thorndike to find the appropriate number of clusters in a dataset. In order to determine the number of clusters, the percentage of variance or Sum of Squared Errors explained by the clusters is plotted against the number of clusters. The Elbow Method proposes that the number of selected clusters should be a number such that adding another cluster does not give much better shaping of the data. This means that the variation in the plot should not be significant. Usually, if the plot of the Sum of Squared Errors *vs* the number of clusters looks like an arm, then the elbow on the arm is the optimal k.

Study the next code, understand what it does and complete the missing lines. Try a little bit on your own before copying the provided solution on gray color.

Code:

```
# import the necessary packages
```

```
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt


# Implementation of the Elbow method
Sum_of_squared_errors = []

#Definition of the number of clusters to evaluated
K = range(1, 30)

for k in K:
    # Training of kmeans model using k clusers
    kmeans_model = KMeans(n_clusters = k, random_state = 42)
    kmeans_model.fit(dataset_standardized)

    # Computing the sum of squared errors for the trained model
    Sum_of_squared_error = kmeans_model.inertia_
    Sum_of_squared_errors.append(Sum_of_squared_error)


# Visualization of the Sum of Squared Error vs the Number of clusters curve
plt.figure(figsize=(10, 5))
plt.plot(K, Sum_of_squared_errors, 'bx-')
plt.xlabel('Number of clusters, k')
plt.ylabel('Sum of squared errors')
plt.title('Elbow Method For Optimal k')
plt.show()
```

Questions

- 2.1.2.1 Describe the relationship between the parameter **k** and the obtained sum of squared errors.
- 2.1.2.2 Based on the results, indicate the optimal parameter **k,** that you consider appropriate to clustering the Airlines Cluster dataset.

2.2. Hierarchical Clustering ***(scipy.cluster.hierarchy.dendrogram, scipy.cluster.hierarchy.linkage)***

(Adapted from https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html)

Hierarchical clustering is an algorithm that builds a hierarchy of clusters. The algorithm begins with all the data assigned to as many clusters as samples. Then, the two closest clusters are joined into a single cluster. This algorithm ends when there is only a single cluster left. The **scipy.cluster.hierarchy** module provides the **linkage()** function to perform the hierarchical clustering in different ways and the **dendrogram()** function to visualize how each cluster is composed.

Building and using the clustering model

(Adapted from https://medium.com/datadriveninvestor/unsupervised-learning-with-python-k-means-and-hierarchical-clustering-f36ceeec919c)

In this section, we are going to use the function **scipy.cluster.hierarchy.linkage()** to perform the hierarchical clustering. This function has the following parameters:

- **y:** corresponds to training vectors.
- **method:** The linkage algorithm used to compute distances between a cluster and a sample. '**single**' corresponds to the Nearest Point Algorithm. '**complete**' is known as the Farthest. Point Algorithm. '**average**' corresponds to the UPGMA Algorithm. '**weighted**' is knows as the WPGMA algorithm. '**centroid**' corresponds to UPGMC algorithm. '**ward**' uses the Ward variance minimization algorithm.
- **metric:** The distance metric to use in the case that *y* is a collection of observation vectors; ignored otherwise.

The **linkage()** function parameters are described in detail in:

https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html#scipy.cluster.hierarchy.linkage

In addition to linkage(), we are going to use the function **scipy.cluster.hierarchy.dendrogram()** to show how each cluster is composed. This function has the next parameters:

- **Z**: The linkage matrix encoding the hierarchical clustering to render as a dendrogram.
- **p:** The *p* parameter to activate the truncate_mode.
- **truncate_mode**: Truncation mode used to condense the dendrogram. '**None'**, no truncation is performed (default). **'lastp'**, the last *p* non-singleton clusters formed in the linkage are shown. "**level**" includes all nodes with *p* (level) merges from the last merge.

The **dendrogram** function parameters are described in detail in:

https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.dendrogram.html#scipy.cluster.hierarchy.dendrogram

Understand the following code and solve the doubts that could come up.

Code:

```
# import necessary packages
from scipy.cluster.hierarchy import linkage, dendrogram
from scipy.cluster.hierarchy import fcluster
from sklearn import preprocessing
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

#Loading and standardizing the data to normal distribution
url = "AirlinesCluster.csv"
airlines_dataset = pd.read_csv(url)
dataset_standardized = preprocessing.scale(airlines_dataset)
dataset_standardized_pd = pd.DataFrame(dataset_standardized)

#Creating the linkage matrix and perform hierarchical clustering on samples
hierarchical_cluster = linkage(dataset_standardized,'ward')

#Plot a complete dendrogram
plt.title('Hierarchical Clustering Dendrogram (complete)')
plt.xlabel('Sample index or (cluster size)')
plt.ylabel('Distance')

dendrogram(hierarchical_cluster,
           leaf_rotation=90,
           leaf_font_size=6,
           )
plt.show()

#Plot a truncated dendrogram at 5 clusters
plt.title('Hierarchical Clustering Dendrogram (truncated)')
plt.xlabel('Sample index or (cluster size)')
plt.ylabel('Distance')
dendrogram(
    hierarchical_cluster,
    truncate_mode='lastp', # show only the last p merged clusters
    p=5,  # show only the last p merged clusters
    leaf_rotation=90.,
    leaf_font_size=12.,
    show_contracted=True, # to get a distribution impression in truncated branches
)
plt.show()


# visualization of the clustered data using 5 clusters
num_clusters=5
hierarchical_cluster_predictions= fcluster(hierarchical_cluster, num_clusters,
criterion='maxclust')
hierarchical_cluster_predictions[0:30:,]

# Plotting clustered data using independent colors
plt.figure(figsize=(10, 8))
```

```
dataset_standardized_pd = pd.DataFrame(dataset_standardized)
plt.scatter(dataset_standardized_pd.iloc[:,0],
dataset_standardized_pd.iloc[:,1],c=hierarchical_cluster_predictions, cmap='prism')
plt.title('Airline Data - Hierarchical Clutering, 5 clusters')
plt.show()

#visualization of descriptive stats of the clustered data using 5 clusters
cluster_Hierarchical_data = pd.DataFrame(hierarchical_cluster_predictions)
airlines_dataset['HierarchicalCluster'] = cluster_Hierarchical_data
hierarchical_cluster_pd =
pd.DataFrame(airlines_dataset.groupby('HierarchicalCluster').mean())
print(hierarchical_cluster_pd)
```

Questions

- 2.2.1 Do you expect that Cluster 1 of the Hierarchical clustering to be necessarily similar to K-Means clustering? Compare the results obtained using K-Means and Hierarchical clustering.
- 2.2.2 Compared to the other hierarchical clusters, Cluster 1 has the largest average values in which features (if any)? Based on this, how would you describe the Airline's customers in Cluster 1? Applied the previous analysis to the remaining four clusters and describe them.
- 2.2.3 Perform a new hierarchical clustering of the AirlineCluster Dataset using a different linkage method and compare with the clustering obtained using the 'ward' method. Explain what happens.

2.3. DBSCAN

(https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html)

Use DBSCAN to cluster the standardized dataset. Cluster the data in three different scenarios, with:

a) eps = 0.3, min_samples = 10
b) eps = 0.3, min_samples = 5
c) eps = 0.2, min_samples = 10

Report the "estimated number of clusters" and the "estimated number of noise points" for each scenario.

Indicate and register in your document:

2.3.1 The estimated number of clusters and noise points in the three previous scenarios
2.3.2 Explain how decreasing "eps" affects to the estimated number of clusters
2.3.3 Explain how decreasing "min_samples" affects to the estimated number of clusters