Exercise 2: K-means Clustering

Introduction

Clustering helps to understand how data samples are related to each other. Clusters are composed by points that are similar to each other, but relatively different to the rest of points. Clustering methods do not need data labels and therefore are called *unsupervised* models, as opposed to supervised ones that we discussed in the previous exercises. In datasets where no labels are available, finding clusters of data points can help grouping the data samples in a principled way.

Objectives

In this exercise you will code the k-means algorithm by prototyping on a synthetic dataset, and then use it to find clusters of world cities based on some of their statistics. You will then analyze the clusters to characterize the different types of cities based on their clustering.

Instructions

Make use of the provided files `Ex_MGI_clustering.R` and `my_kmeans.R` and fill in the gaps by following the instructions in this document. In the gray boxes, **R** means R coding action required and **Q** points at questions to answer.

Tasks

1 Write your k-means algorithm and test it on a 2D dataset

We have seen how k-means works during the lectures. You will first write the core of the k-means loop in the main exercise file `Ex_clustering.R`, then extend it in the function `my_kmeans.R` (skeleton file provided).

1.1 Initialize the cluster centroids

Load `data/data2d.csv` in your workspace as `Data2D`. It contains data points of dimensionality 2, in which each row is a point. It has been built by sampling data points from a Gaussian mixture model (i.e. a sum of Gaussian distributions). In order to work, k-means needs to start positioning the centroids somewhere in an initialization step. The easiest way is to randomly select K points from the dataset.

- **R**1: Initialize the cluster centroids. Choose K points randomly from the dataset to do this. This can be done by randomly selecting K indexes between 1 and the number of rows in `Data2D` and them taking the corresponding rows from `Data2D`. You are going to find the function `sample()` very useful for this. Check if you need to change the value of the option `replace`. Remember that the number of rows can be obtained with the function `nrow()`.
- **Q**1: Why do we randomly select existing data points for the initialization, instead of just assigning random values?

1.2 Assign each data point to the nearest cluster centroid

These random centroids are, probably, not very good. The first step for improving them iteratively is to find out which is the nearest centroid for each element in `data`. You can see this step as assigning each data point to the cluster defined by each centroid.

- **R**2: Compute a vector of indexes which has as many elements as the number of data points and that can take a value from 1 to K (the cluster index). This index specifies which is the nearest centroid for each data point. Call this vector `assigned_clusterIDs`. You can use the function `distmat()`, either from the libray `pracma` or from the provided file `distmat.R` by loading it with the function `source()`. `distmat()` computes the distances between centroids and data points. It might require to use `as.matrix()` on both inputs. Then use the output of the `which.min()` function to find the index of the closest centroid to each data point. The function `apply()` can help you doing this in a single line of code. Look online how `apply()` works, since you might need it several times in this exercise.
- **Q**2: Use the provided code to plot the results. If you haven't used any data normalization, you might find surprising the centroid assignment. Why do you think is that?
- **R**1: Improve this effect using the `scale()` function or any other way of normalizing the data and compute everything again.

1.3 Update the centroids

Now compute the new values for the centroids.

- **R**3: Write a `for` loop, from 1 to K, and update each cluster centroid with the actual mean value of all the samples assigned to it. This can be done with the function `mean()`. Note that, if `assigned_clusterIDs` contains the cluster assignments, then the indexes of the data samples can be retrieved by a comparison operator (*e.g.* $>$, $<$, $!=$ or $==$).

1.4 Build your k-means function and compare it to R's k-means

- **R**4: Use the code in `YOUR CODE HERE` sections into the `my_kmeans.R` function. Make sure that you embed your code properly in the loop that controls the iterations, running from 1 to `maxiter`.
- **Q**3: Study the process across iterations. Why do cluster centers move during iterations? When do they stop being updated and why?

2 Visually explore the OECD cities dataset

Open the `oecd_cities_stats.xlsx` file using Excel. In it, you will find 8 features of metropolitan areas from some OECD countries obtained from *stats.oecd.org*. Now open the file using the code provided and take a look at the data frame *DataCities*. You want to understand how the cities can be grouped together and how each geographic region differs from the others. We provide you with three lines of code to plot a matrix with all variable pairs as a first way of taking a look at the data. Since it's a bit tedious to look at the data like this, we will try to summarize it using two unsupervised algorithms: k-means clustering and PCA.

- **R**5: Apply the k-means algorithm to this dataset with $k = 3$ to $k = 5$, with and without normalizing the data with `scale()`. We provide the code for visualizing the cluster centroids.
- **R**6: Use the `RegionNames` variable to check which world regions fall in each cluster. This can be done using the `pie()` function, for plotting pie charts, and the `table()` function for counting occurrences.
- **Q**4: What are the main characteristics of these 3 to 5 groups in the two cases? What difference does normalization do? Describe a bit each region and each cluster based on what you observe. Note that k-means is helping you to summarize the data by grouping the cities according to their variables.
- **R**7: Use the `prcomp` function to perform Principal Component Analysis on the data. Then use the provided `ggbiplot` function to visually explore the results by plotting several pairs of principal components (for instance with the option `choices=c(1,2)`, `choices=c(1,3)` and `choices=c(2,3)`).
- **Q**5: How would you describe each of the first 3 PCA components in terms of the 8 original variables? Which regions dominate each of the components? Does it provide more information to plot further principal components? Describe each region based on this and compare with the k-means analysis. Note that PCA presents you with groupings of variables based on the cities that have them.