Learning from Big Data: Module 1 - Natural Language Processing

Session 1 - getting familiar with unsupervised learning

7/18/2022

Introduction

This is a first, rather simple, script that has the goals of (1) getting you familiar with R (if you are not yet so) and (2) having you to black-box run a very simple unsupervised learning task. In fact, you will be running what is arguably the simplest unsupervised learning task - a cluster analysis to group observations based on their similarity. For now, this is done by calling the function kmeans, from the library cluster (for documentation, see https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/kmeans)

In clustering, our goal is to maximize within-cluster homogeneity while making the clusters as different as possible; in other words, we wish to find the set of clusters such that observations within each cluster are as similar to each other as possible, while the clusters (more specifically, their means) are as different from each other as possible. This is a classical example of unsupervised learning.

For now, I will not ask you to "open the black box" and explain how this specific unsupervised algorithm works. We will have time to explore the mechanics and details of specific NLP-appropriate supervised- and unsupervised- learning algorithms in the coming lectures and assignments.

Loading libraries

Before starting, make sure you have all the required libraries properly installed. Simply run this chunk of code below.

```
# Packets required for subsequent analysis. P_load ensures these will be installed and loaded.
if (!require("pacman")) install.packages("pacman")
pacman::p_load(ClusterR, cluster  )
```

1. Load and prepare the data

We will use the famous Iris dataset (https://en.wikipedia.org/wiki/Iris_flower_data_set) because it is simple, appropriate for classification, and comes built-in into any R installation. In the next tutorials, we will be loading the data ourselves, typically by reading csv files (using read.csv()).

The dataset has basically four variables: the length and width of the flower sepal and petal. The goal is to use these variables to identify the species.

```
data(iris);str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
```

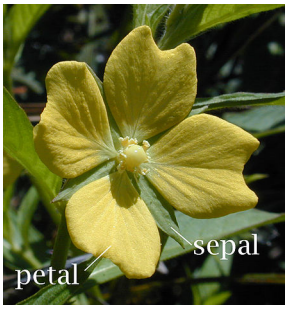


Figure 1: Sepal and Petal.Source: wikipedia

```
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
iris_no_sp <- iris[, -5] # Remove the species label from the data
```

2. Fit a K-Means clustering model to the training dataset

Now we will fit a clustering model using the K-Means algorithm. Note that this requires us to specify, a priori, the number of clusters we want the algorithm to produce. In this example case we set it at 3.

```
set.seed(123) # Setting seed
kmeans.result <- kmeans(iris_no_sp, centers = 3,    nstart = 20)
```

Here are some summary statistics in the three identified clusters :

```
kmeans.result
```

```
## K-means clustering with 3 clusters of sizes 50, 62, 38
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     5.006000    3.428000     1.462000    0.246000
## 2     5.901613    2.748387     4.393548    1.433871
## 3     6.850000    3.073684     5.742105    2.071053
##
## Clustering vector:
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3
## [112] 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [149] 3 2
##
## Within cluster sum of squares by cluster:
## [1] 15.15100 39.82097 23.87947
##  (between_SS / total_SS =  88.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

3. Identify the cluster for each observation

```
kmeans.result$cluster
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3
## [112] 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 3 2 3 3 3 2 3 3 3 2 3
## [149] 3 2
```

4. How well does the classification do?

To check the performance of our algorithm, we compare the clusters against the ground truth, i.e., the actual species, which is found in the variable 'species' in the dataset. Note that we have not used the variable 'species' so far; we only use the dimensions of the sepal and petal to create the clusters; now we will see how much hit and how much miss did we get in this task.

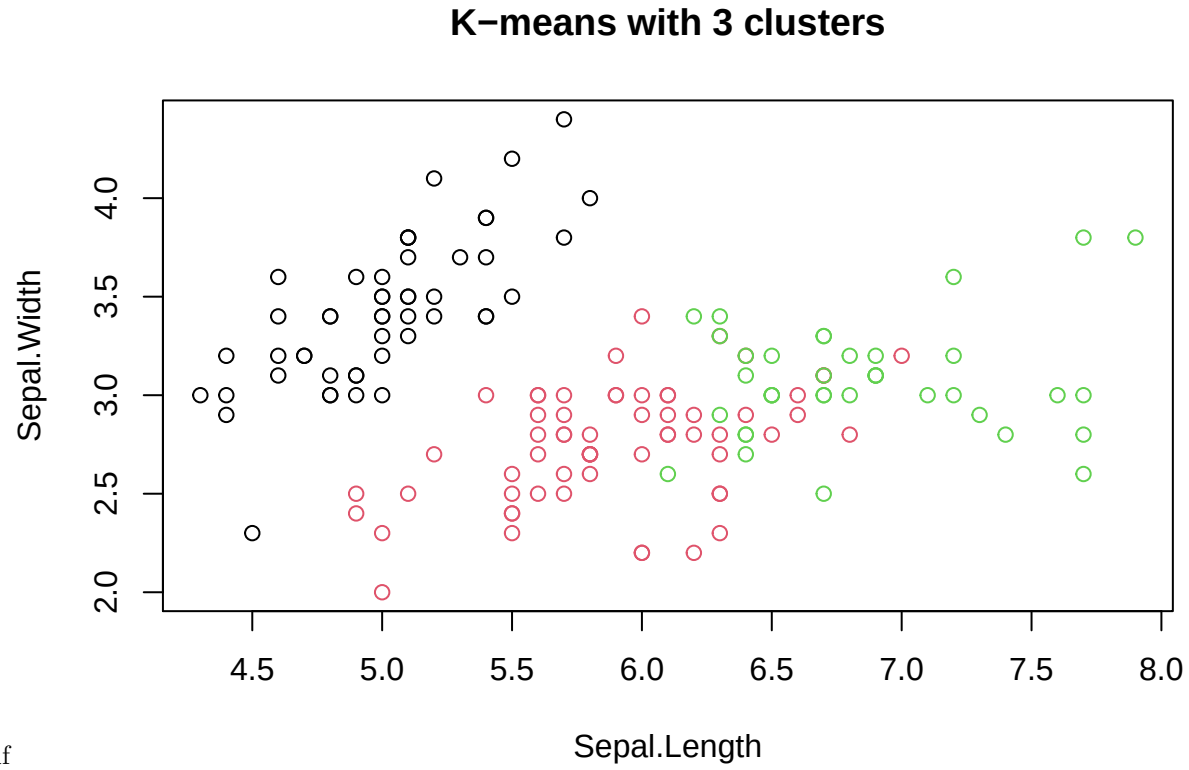
```
confus <- table(iris$Species, kmeans.result$cluster)
confus
```

```
##
##                1  2  3
##   setosa      50  0  0
##   versicolor   0 48  2
##   virginica    0 14 36
```

Before we finalize, do note that this is an extremely simplified view of a classification task. In this course, we will discuss and practice important things that were not done in this first exercise, such as separation of samples for training and validation, and cross-validation. Thus, take these results above with a grain of salt. It is good to be skeptical since day 1 :)

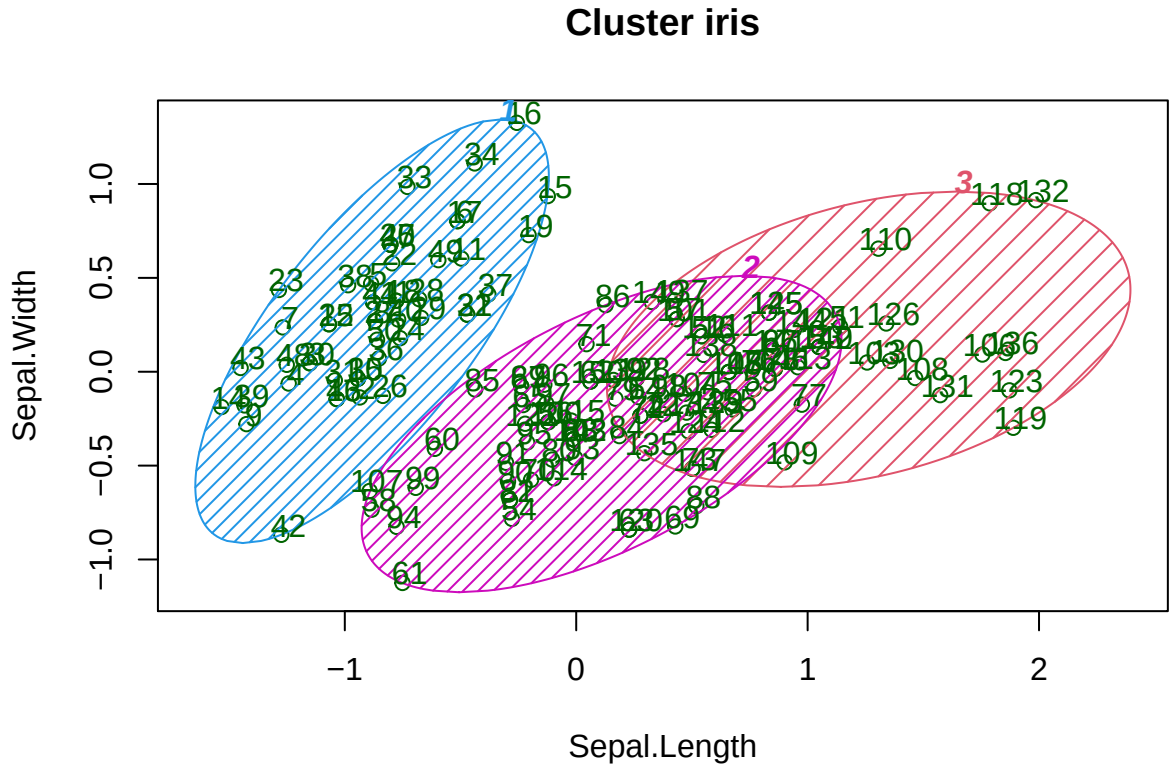
5. Let's look at the clusters

```
plot(iris_no_sp[c("Sepal.Length", "Sepal.Width")],
     col = kmeans.result$cluster,main = "K-means with 3 clusters")
```



clusters-1.pdf

```
y_kmeans <- kmeans.result$cluster
clusplot(iris_no_sp[, c("Sepal.Length", "Sepal.Width")],
         y_kmeans, lines = 0, shade = TRUE, color = TRUE, labels = 2,
         plotchar = FALSE, span = TRUE,     main = paste("Cluster iris"),
         xlab = 'Sepal.Length', ylab = 'Sepal.Width')
```



clusters-2.pdf