Global complexity versus Elbow method to choose the hyper-parameter k of k-means

Ghribi Saif Eddine
Supervised by
Mr Jean-Louis Dessalles
Mr Pierre-Alexandre Murena

March 23, 2019

Contents

1 Introduction

During the Athens week and thanks to the subject From Complexity to Intelligence (TPT37),I was introduced to the theory of complexity and its relation with machine learning.In the lab of Machine Learning of this subject the idea of choosing the optimal number of clusters with global complexity was well introduced and explained.Thus,since I already worked with k-means in several projects I have always used the elbow method to determine this hyper-parameter.In this short report I will explain and compare these two methods.

2 Data used in the project

In order to compare the two methods I have chosen to use 4 of the widely used data sets in machine learning for testing. These data sets will be all retrieved from SKlearn .

- Sklearn Iris : This data sets consists of 3 different types of irises.
- Sklearn Make Moons : Make two interleaving half circles.
- Sklearn Make blobs : Generate isotropic Gaussian blobs for clustering.
- Sklearn Wine data :The wine dataset is a classic and very easy multi-class classification dataset.

To store the data for further use I created a dictionary in a function called data to retrieve data by its name:

```
def data(index):
  dictionary = {}  # dictionary to store data
  d = load_iris()
  X = d['data']
  dictionary["iris"] = X  # add iris data (index "iris")
  X, y = make_moons(noise=.05, random_state=1)
  dictionary["moons"] = X
  X, y = make_blobs(random_state=1)
  dictionary["blobs"] = X
  d = load_wine()
  dictionary["wine"] = d['data']
  return dictionary[index]
```

Listing 1: Function to retrieve data

3 Elbow method

3.1 Overview

The elbow method is used to determine the optimal number of clusters in k-means.
if k increases, average distortion will decrease, each cluster will have fewer constituent instances, and the instances will be closer to their respective centroids.
The value of k at which improvement in distortion declines the most is called the elbow, at which we should stop dividing the data into further clusters.
The distortion function here is considered as the sum of squared distances of samples to their closest cluster center.

3.2 Example using Iris data set

In order to plot the elbow method I used a list to store the sum of squared distances of samples to their closest cluster center for a set of integers ,a set of the number of clusters.

```
# fix the figure size
figure(num=None, figsize=(8, 6), dpi=90, facecolor='w', edgecolor='k')
l_elbow, l_kolo = eval_complex(max_clusters, "iris")
plt.plot(range(1, max_clusters), l_elbow, label="elbow method")
plt.plot(6, l_elbow[5], "o", label="elbow point = 6")
plt.xlabel("Number of clusters")
plt.ylabel("Sum of squared distances of samples to their closest cluster center")
plt.title("Elbow method")
plt.legend()
```

Listing 2: Plot Elbow method

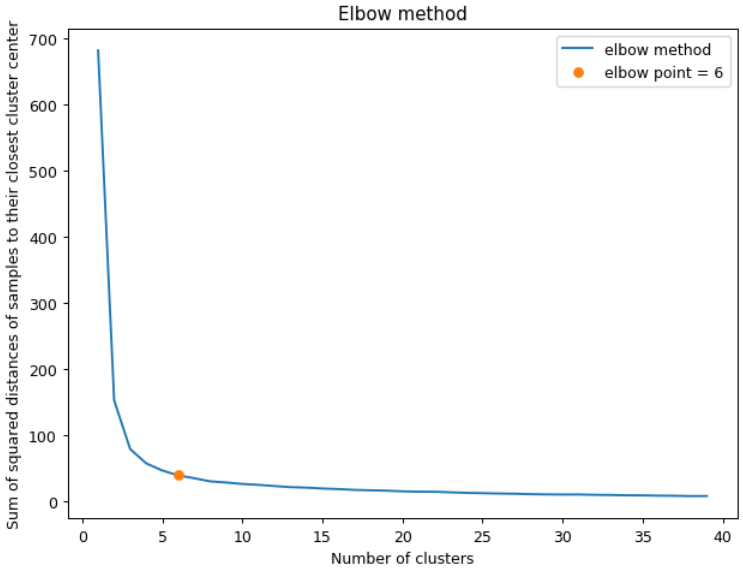

Figure 1: Elbow method

4 Global Complexity

4.1 Definition

The Global complexity was first introduced in the fourth lab.The definition of the global complexity is :

$$GlobalComplexity = Complexity(Data|Protoypes) + Complexity(Protoypes) \quad (1)$$

The prototypes here are the centers of clusters known us centroids. The function that determines this complexity is already implemented by Mr Jean-Louis Dessalles and Mr Pierre-Alexandre Murena as follows :

4.2 Implementation

```
def Cmatrix(M):
  return np.sum(np.log2(1 + abs_app(M)))

def gradCmatrix(M):
  return np.sign(M) / (1.0 + abs_app(M))

def C_prototypes(M):
  return Cmatrix(M) + np.log2(M.shape[0])

```

```
def C_data_knowing_prototypes(D, P):
  NearestPrototype = associatePointToPrototype(D, P)
  return Cmatrix(D - P[NearestPrototype,:])

def globalComplexity_clustering(X, P):
  return C_data_knowing_prototypes(X, P) + C_prototypes(P)
```

Listing 3: Global Complexity implemented by Mr Jean-Louis Dessalles and Mr Pierre-Alexandre Murena

5 Elbow method and Global Complexity comparison

5.1 The main idea

To compare these two methods one shall plot the reaction of the Global Complexity when the hyper-parameter k varies and the distortion function which is the sum of squared distances of samples to their closest cluster center.

5.2 Implementation

5.2.1 Function to evaluate Global Complexity and the distortion

```
def eval_complex(max_clusters, data_index):
  # list to store Sum of squared distances of samples to their closest cluster
    center.
  l_elbow = []
  # list to store the general complexity
  l_kolo = []
  for i in range(1, max_clusters):
  # choose data
  X = data(data_index)
  # fit model (k-means)
  kmeans = KMeans(n_clusters=i, random_state=1).fit(X)
  # global complexity after K-Means using the function
    compareComplexitiesWithKMeans(Data,K) from the lab session
  Cmatrix, global_complexity = compareComplexity(X, i)
  # store the values in corresponding lists
  l_elbow.append(kmeans.inertia_)
  l_kolo.append(global_complexity)
  return l_elbow, l_kolo
```

Listing 4: Function to evaluate Global Complexity and the distortion

5.2.2 Function to plot the comparison for a given data name

```
def plot(max_clusters, data_name):
  # retrieve the two lists of elbow method and global complexity to plot
  l_elbow, l_kolo = eval_complex(max_clusters, data_name)
  # fix the figure size
  fig, ax1 = plt.subplots(figsize=(12, 8))
  # use subplot to show the two plots according to their scales.
  # plot the global complexity
  ax1.set_xlabel("Number of clusters K")
  ax1.set_ylabel('Global Complexity', color='red')
  ax1.plot(range(1, max_clusters), l_kolo, color="red")
  # plot the minimum of global complexity
  ax1.plot(np.argmin(l_kolo)+1, l_kolo[np.argmin(l_kolo)],
  "o", label="min complexity="+str(np.argmin(l_kolo)+1))
  ax1.legend()
  # plot the elbow method which is Sum of squared distances of samples to their
    closest cluster center.
```

```
  ax2 = ax1.twinx()
  ax2.set_ylabel(' Elbow Method', color='green')
  ax2.plot(range(1, max_clusters), l_elbow, color="green")
  # plot the title of the plot accordinf the data used
  fig.suptitle("Data:"+data_name, size=16)
  fig.tight_layout()
  fig.subplots_adjust(top=0.90)
```

Listing 5: Function to plot the comparison for a given data name

5.3 Results for the four data sets

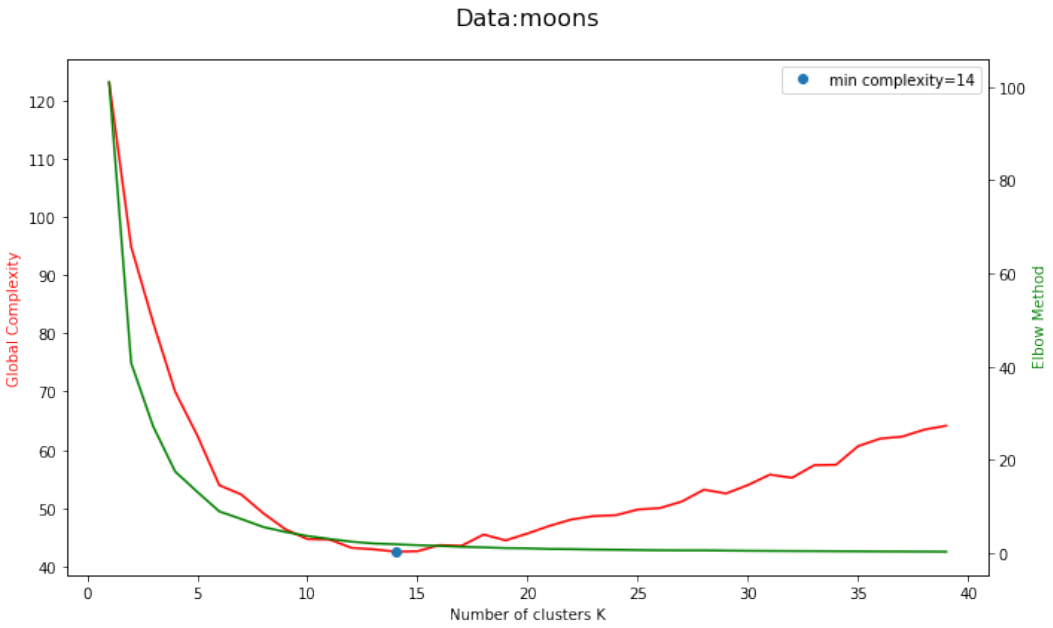

Figure 2: Elbow method vs Global Complexity (Moons data set)

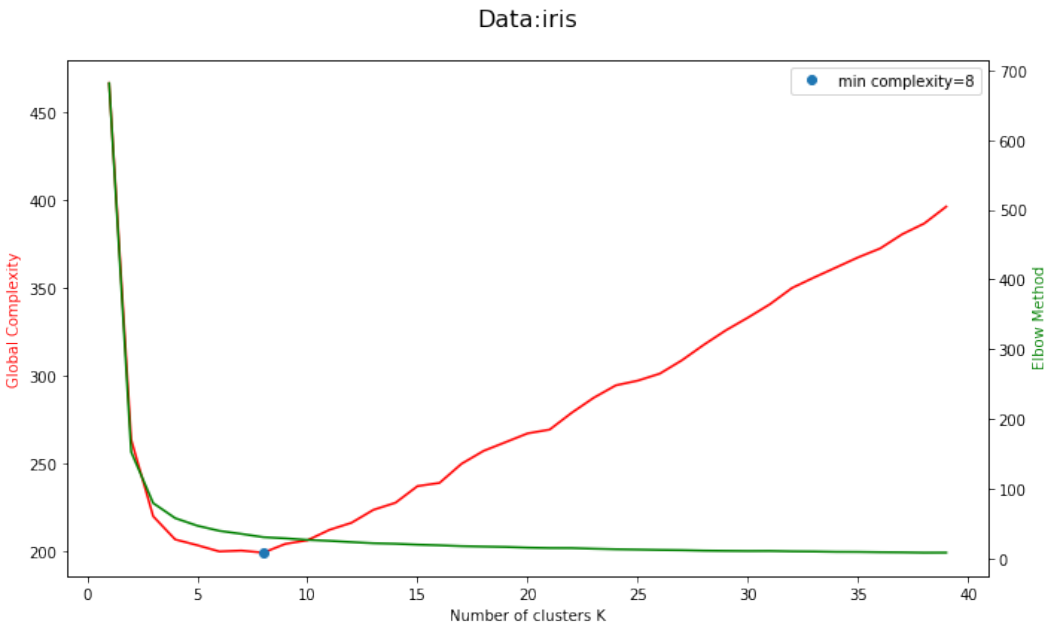

Figure 3: Elbow method vs Global Complexity (Iris data set)

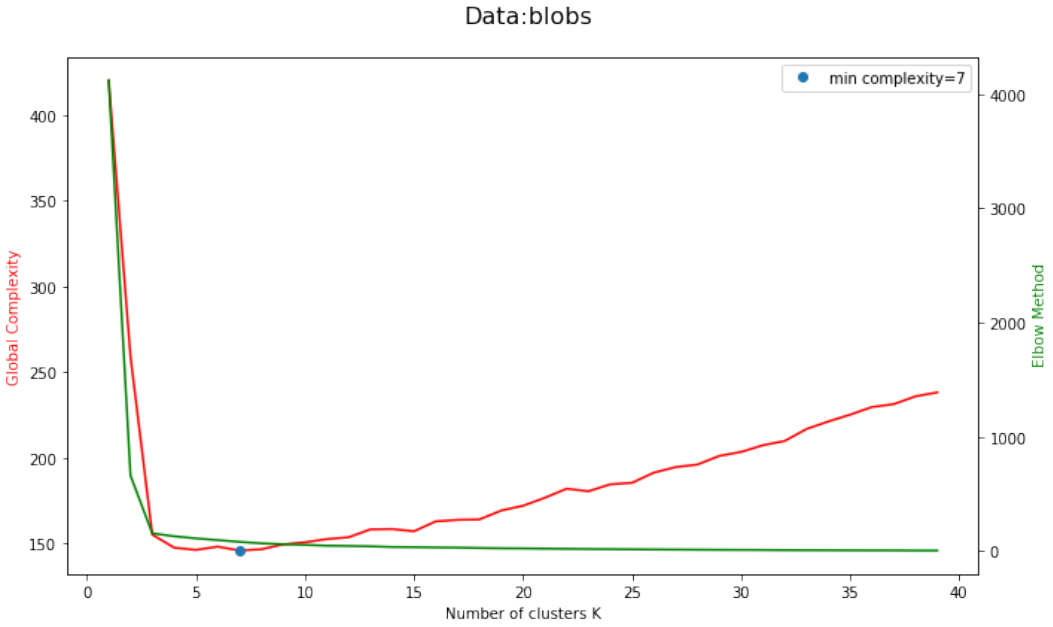

Figure 4: Elbow method vs Global Complexity (Blobs data set)

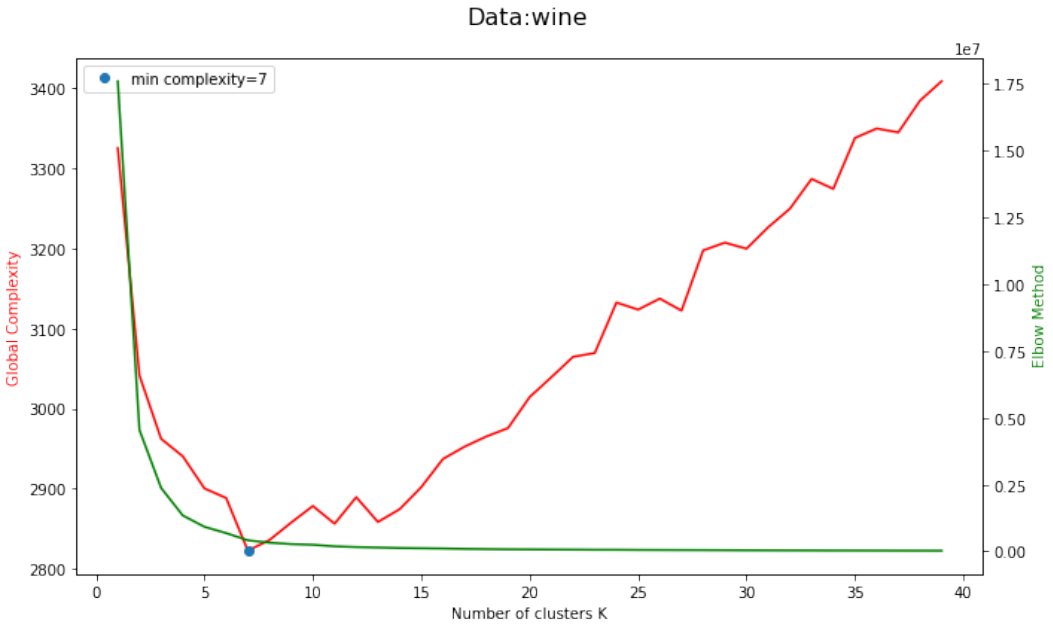

Figure 5: Elbow method vs Global Complexity (Wine data set)

5.4 Commentary

The results are certainly fascinating the Argmin of the Global Complexity is so close to what we choose manually with the elbow method.We notice also that for the four data sets the global complexity is always a strict convex function which will come very handy if we want to have a unique optimal point, a unique number of clusters.

6 Conclusion

Unsupervised learning is a very effective branch of machine learning.Nevertheless, the hyper-parameters in unsupervised learning like the number of clusters in k-means and the number of hidden layers in neural networks in supervised learning for example are one of the biggest challenges for machine learning experts.The complexity here surprisingly explained very well how the complexity decreases if we approach the optimal number of clusters.Thus,a more formal research about this topic would lead to brilliant results.

7 References

The implementation of global complexity used in this project is implemented by Mr Jean-Louis Dessalles and Mr Pierre-Alexandre Murena .You can contact them for further information.