Bishop's University

CS 405/CS 505 – Data Mining

Assignment 3: K-means

1. Introduction

The objective of this assignment is to present the use of scikit-learn functionalities concerning automatic classification with k-means, as well as to contribute to a better understanding of this method and the impact on the results of the data distribution or initialization technique (random initialization or k-means++). For this, data generated in a controlled manner are first examined and then real data seen in progress, with or without pre-processing to reinforce the separation between groups.

2. Automatic classification in scikit-learn

Various automatic classification methods are available in scikit-learn. Among these methods we will examine more closely the K-means (K-means) and later the spectral classification (spectral clustering).

For each method, the implementation allows to obtain a "model" (e.g., a set of k cluster centers) from the data (as an array of `n_samples x n_features` as well as, for some methods, a matrix of similarities `n_samples x n_samples`) and then to obtain the group number (cluster) for the initial data or for new data.

The description of the implementation of the K-means method can be found in this link. We will also use the adjusted Rand index, to assess the consistency between different classifications of the same set of data.

3. Classification with K-means of generated data

Start by generating five groups of 100 vectors each in 3D space, each following a normal distribution (with zero mean and unit variance). Apply to each group a different translation in space, generate the group labels, build the total data set and then shuffle the rows of this set:

```
import numpy as np
from sklearn.utils import shuffle

# generation of 100 3D points according to the centered normal law
# each group is translated by a vector [3,3,3]
d1 = np.random.randn(100,3) + [3,3,3]
d2 = np.random.randn(100,3) + [-3,-3,-3]
d3 = np.random.randn(100,3) + [-3,3,3]
d4 = np.random.randn(100,3) + [-3,-3,3]
d5 = np.random.randn(100,3) + [3,3,-3]

# generation of labels for each group
c1 = np.ones(100)
c2 = 2 * np.ones(100)
c3 = 3 * np.ones(100)
c4 = 4 * np.ones(100)
c5 = 5 * np.ones(100)

# concatenation of data in a matrix
data = np.concatenate((d1,d2,d3,d4,d5))
labels = np.concatenate((c1, c2, c3, c4, c5))
print(data.shape)
# random permutation of the rows of the data matrix
data, labels = shuffle(data, labels)
```

Visualize the starting groups:

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
# The color of the points depends on their label
ax.scatter(data[:,0], data[:,1], data[:,2], c=labels)
plt.show()
```

Apply automatic classification with K-means, first with a single trial (a single initialization followed by a single execution of K-means, `n_init=1`) using the `k-means++` initialization method:

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=5, n_init=1, init='k-means++').fit(data)
```

Review the parameters, attributes, and methods of the `sklearn.cluster.KMeans` class by following the link given above. The predicted groups for the data can be obtained using the `predict(X)` method:

```
pred = kmeans.predict(data)
```

Groups associated with training examples are also stored in the `kmeans.labels_` attribute:

```
print(kmeans.labels_)
```

Visualize the results of this classification:

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(data[:,0], data[:,1], data[:,2], c=pred)
plt.show()
```

It is possible to assess the consistency between the starting groups and the partitioning found by K-means using the adjusted Rand index:

```
from sklearn import metrics
metrics.adjusted_rand_score(pred, labels)
```

The call to `metrics.adjusted_rand_score()` compares the partitioning obtained by the automatic classification (pred group labels) with the partitioning corresponding to the groups defined at the start (labels stored in labels).

Now apply automatic classification with K-means with a single trial (`n_init = 1`) using the random initialization method:

```
kmeans = KMeans(n_clusters=5, n_init=1, init='random').fit(data)
metrics.adjusted_rand_score(kmeans.labels_, labels)
```

Question 1: Repeat the classification several times with each of these two initialization methods and each time examine the consistency of the groups obtained with the starting groups. What do you notice? Explain.

Question 2: Vary the number of clusters (`n_clusters`) and experiment for each value of the number of clusters. Re-examine the stability of the results using the adjusted Rand index. Explain what you see.

Question 3: Vary the number of groups (`n_clusters`) between 2 and 20, plot the evolution graph of the final value reached by the cost (inertia, see documentation) for each of the values of `n_clusters`.

Question 4: Generate 500 data with a uniform distribution in $[0,1]^3$ (three-dimensional data in the unit cube). Apply on these data K-means with `n_clusters=5` and random initialization (random) and examine the stability of the results using the Rand index. Apply on this same K-means data with always `n_clusters=5` but a `k-means++` initialization, examine the stability of the results. Be careful, you no longer have any groups defined at the start; to define the reference groups, to which you will compare those from other classifications, you can apply K-means for the first time with `n_clusters=5, n_init=1, init='k-means++'`. Do you observe any differences compared to the results obtained on the data generated at the beginning of this section (with `np.random.randn`)? Explain.

4. Classification with K-means of "texture" data

As a reminder, these data correspond to 5500 observations described by 40 variables. Each observation belongs to one of the 11 texture classes; each class is represented by 500 observations. We will apply K-means to this data, with `n_clusters = 11`, and examine how close the groups resulting from the automatic classification are to the present classes.

Mix the observations and apply K-means to this data (be careful, the last column contains the class labels):

```
textures = np.loadtxt('texture.dat')
np.random.shuffle(textures)
kmeans = KMeans(n_clusters=11).fit(textures[:,:40])
metrics.adjusted_rand_score(kmeans.labels_, textures[:,40])
```

The groups resulting from the automatic classification give only a few indications on the classes present in these data.

Question 5: Apply discriminant analysis to this data and again apply K-means with `n_clusters = 11` to the projected data in the discriminant space. What do you notice? Explain. Visualize the results.

5. Submission

You must submit the *pdf* file of your report. In fact, it must contain the responses for the eleven questions. In addition, you have to submit your *.ipynb*. ***Please, do not submit your assignment in .zip or .rar files.***

6. Useful external references

- NumPy documentation
- SciPy documentation
- MatPlotLib documentation
- scikit-learn
- Python programming language