Bishop's University

CS 405/CS 505 – Data Mining

Final Exam: DBSCAN

The exam is worth 100 points total. Be sure to read the whole exam before attempting any of it. This exam is open book since it is online. ***Please note that this exam is individual and not in group****. The exam is at December 11th, 2022 from 8:30 am to 11:59 pm.*

Submission: *All submissions must be performed through Moodle. The exam is from 8:30 am to 11:59 pm. Any submission with errors will get attributed the grade zero. No submission by email will be accepted. Please note that any late* ***submission or update*** *will involve a penalization of 10% for 30 minutes late submission, 20% of one hour, etc.* ***The penalization will be applied for submissions or updates after 00:15 pm.*** *It will be of your interest to be efficient in managing your time.*

1. Introduction

The objective of this final exam is to illustrate the practical application of automatic classification with DBSCAN. First, we will observe the behavior of the algorithm in a synthetic case involving k-means failure. We will see how to use classical heuristics to determine the parameters of the algorithm and how to interpret the partitioning obtained. Finally, we will apply DBSCAN on a real data set and we will illustrate its use for the detection of outliers.

2. Density classification in scikit-learn

As we saw in the assignments, scikit-learn offers many automatic classification methods. For this final exam, we are interested in DBSCAN (Density-Based Spatial Clustering of Applications with Noise).

Like k-means, DBSCAN makes it possible to obtain for an observation matrix `n_samples x n_features` a vector corresponding to the `n_samples` numbers of the groups (cluster) for these observations. On the other hand, DBSCAN does not make it possible to produce classifications for new data (it is necessary to retrain the model by including new observations). The description of the DBSCAN implementation can be found in https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html.

3. Classification with DBSCAN of generated data

Let's start by generating a dataset of 500 observations in 2D space. We will use a function built into scikit-learn to produce circular point clouds.

```
import matplotlib.pyplot as plt
import numpy as np
import sklearn.datasets
from sklearn.utils import shuffle

# Let's generate a scatter plot composed of two circles
# The cloud contains 500 observations (`n_samples`) noisy by
# a Gaussian noise of standard deviation 0.1 (`noise`).
# The ratio between the radius of the small circle and the large circle
# is 0.3 (`factor`).
data,  labels  =  sklearn.datasets.make_circles(n_samples=500,  noise=0.1,
factor=0.3, random_state=0)

print(data.shape)
# Random permutation of the rows of the matrix (we mix the observations)
data, labels = shuffle(data, labels)

# Point cloud display
plt.scatter(data[:,0], data[:,1], c=labels)
plt.show()
```

Question 1: How many groups does this dataset have?

Question 2: Perform a clustering of this dataset using k-means. What can we expect? What do you notice?

Since the two circles are separated by an area with no data, a density-based method seems appropriate. We can create a clustering model using DBSCAN by importing it from scikit-learn:

```
from sklearn.cluster import DBSCAN

db = DBSCAN()
```

The constructor arguments of DBSCAN are as follows:

- `eps`: the dimension of the neighborhood, i.e. the maximum distance between two observations allowing them to be considered as neighbors of each other,
- `min_samples`: the minimum number of neighbors that a central point must have,
- `metric`: the distance to consider (by default, the Euclidean distance is used).

You can call the following methods:

- `.fit(X)`: performs an automatic classification using the DBSCAN method on the observation matrix X. The results are stored in the `.labels_` attribute.
- `.fit_predict(X)`: same as `.fit(X)` but returns group labels directly.

The following attributes are available after calling the `.fit()` method:

- `core_sample_indices_`: the indices of the core points.
- `labels_`: the group numbers of the points in the observation matrix.

Question 3: What are the default values for important DBSCAN parameters in scikit-learn (ε and m)?

Let apply an automatic classification by DBSCAN on our dataset. As for k-means, it is possible to perform this step in two steps by calling `fit()` then accessing the `labels_` attribute, or to do everything in one operation using the `fit_predict()` method

```
predictions = db.fit_predict(data)
# equivalent to
# db.fit(data)
# predictions = db.labels_

# Display of the scatter plot colored by the predictions
```

```
plt.scatter(data[:,0], data[:,1], c=predictions)
plt.show()
```

Question 4: What do you notice? On what parameter is it probably necessary to play to improve this result?

To refine our analysis, we will apply Schubert's heuristic, which exploits the k-distance graph in the observation cloud. We consider for the moment that `min_samples` is fixed at its default value, i.e. 5. We must therefore draw the graph of the 4-distances for our observation matrix.

```
from sklearn.neighbors import NearestNeighbors

nn = NearestNeighbors(n_neighbors=4).fit(data)
distances, _ = nn.kneighbors(data)
```

Question 5: Using the NearestNeighbours documentation in scikit-learn, explain what the code above does.

We can now draw the 4-distance graph. To do this, we only keep the distance from each point to its fourth neighbor, then we sort this list in descending order.

```
distances_triees = np.sort(distances[:,-1])[::-1]
plt.plot(distances_triees)
plt.xlabel("Number of points")
plt.ylabel("$4$-distance")
plt.show()
```

Question 6:

From the 4-distance graph, determine the appropriate `eps` value for this dataset using the current view heuristic. Reapply DBSCAN with these settings. Display the resulting point cloud.

Question 7: How many groups do you get? What are observations with label -1?

4. Classification with DBSCAN of "Iris" dataset

The Iris dataset is a classic in statistical learning. It includes 150 observations of plants according to four attributes:

- sepal length,
- sepal width,
- petal length,
- width of the petal.

Sightings fall into one of three classes, corresponding to the three species of iris: Setosa, Versicolour, or Virginica. More details can be found in the documentation. Iris is built into scikit-learn and is available from the sklearn.datasets submodule:

```
from sklearn.datasets import load_iris

X, y = load_iris(return_X_y=True)
```

Question 8: How many observations does this dataset contain?

To simulate the presence of outliers, we will randomly generate 20 noisy points, drawn according to a uniform law between the minimum and maximum values of each column of the observation matrix:

Question 9: Perform a principal component analysis and visualize the Iris dataset projected along its first two principal axes.

Answer: (I provided you the answer to be able to do the question 9)

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
plt.scatter(X_pca[:,0], X_pca[:,1], c=y) and plt.show()
```

Question 10: Apply automatic classification using DBSCAN (work on 4-dimensional data, not projected data!). Visualize the groups obtained in the main plan. Compare this result to the partitioning obtained using a k-means.

The points corresponding to the label -1 are those that have been identified as outliers. If the value of ε has been well chosen, the noisy observations that we have injected into the dataset should thus have been isolated by DBSCAN.

Question 11: Using the functions of sklearn.metrics, calculate the good detection rate of outliers. Up to what proportion of noisy data is the partitioning obtained by DBSCAN robust?

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 Final exam in .zip or .rar files.***

6. Useful external references

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