EM v.s. DBSCAN

Miguel Alcón Doganoc
Universitat Politècnica de Catalunya
Barcelona, Spain
miguel.alcon@est.fib.upc.edu

1 INTRODUCTION

Clustering algorithms are really relevant in the scientific community. They are used in several fields like biology, medicine, business and marketing, computer science, etc.

Because of that, we want to analyze and compare two of the most well-known algorithms in cluster analysis, which are Expectation-maximization (EM) and Density-based spatial clustering of applications with noise (DBSCAN). We chose both algorithms because they have really different ideas behind its implementation, so we expect that they are significantly different to be compared.

To be more precise, the main objective of our work is to apply different datasets to both algorithms, analyze the results and their differences to select a winner, if possible.

2 ALGORITHMS

2.1 EM

Expectation-maximization algorithm (EM) [**?** **?** **?**] is an iterative method to find maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables. The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step.

As a clustering algorithm, given a fixed k clusters, EM computes probabilities of cluster memberships based on one or more probability distributions. Put another way, the EM algorithm attempts to approximate the observed distributions of values based on mixtures of different distributions in different clusters. The goal of the clustering algorithm then is to maximize the overall probability or likelihood of the data, given the (k) clusters.

The results of EM clustering are the classification probabilities of each observation of the data. In other words, each observation belongs to each cluster with a certain probability. Of course, as a final result you can assign observations to clusters, based on the (largest) classification probability.

2.2 DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) [**?** **?**] is a data clustering algorithm. Specifically, it is a density-based clustering non-parametric algorithm. So, given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away). This algorithm require two parameters:

- ε: the maximum distance between two points for them to be considered as in the same neighborhood.
- $minPts$: the minimum number of points required to form a dense region.

Given ε and $minPts$, the DBSCAN algorithm can be abstracted into the following steps:

(1) Find the points in the ε neighborhood of every point, and identify the core points with more than $minPts$ neighbors.
(2) Find the connected components of core points on the neighbor graph, ignoring all non-core points.
(3) Assign each non-core point to a nearby cluster if the cluster is an ε neighbor, otherwise assign it to noise.

3 DATA

In order to perform the experimentation and to compare both clustering algorithms, we created different scenarios using the module `dataset` of the *scikit-kearn* [**?**] package of Python. These scenarios consist in 2-D points located in different places among the space, forming clusters of different shapes (blob, circle and moon). They are shown in figure **??**.

We also used the dataset of the previous work (Data mining techniques applied to medicine), which is the Heart Disease dataset [**?**], to see how the algorithms behave with realistic data.

4 IMPLEMENTATION

All the experiments were programmed with Python 3. We used the well-know *scikit-kearn* [**?**] Python package to apply the clustering methods and to calculate their accuracy (with the adjusted random score), and *matplotlib* [**?**] to create the plots of the first and second experiments.

For more details of the implementation, you can always take a look at the Python script that we developed (inside *'src'* folder).

5 EXPERIMENTATION

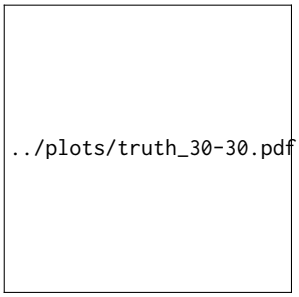
5.1 First experiment

The goal of the first experiment is to observe the differences of applying EM and DBSCAN to the generated data with blob shape. For this, we created an algorithm that given a number of blobs (n_b) and the space between them (s_b), it places the n_b isotropic Gaussian blobs in a well-distributed way, and with centers separated at a distance of s_b (see table **??**). We tried EM and DBSCAN for several cases generated with the algorithm, but we selected 4 of them that show clearly the differences between both. These are represented in sub-figures **??**, **??**, **??** and **??**.

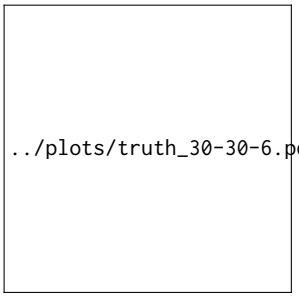
The selected parameters for EM (k) vary for each dataset, but for DBSCAN we fixed them to $\varepsilon = 0.2$ and $minPts = 10$, with which we obtained better results after trying several possibilities. Notice that the colors of the clusters in figures **??** and **??** are not an identifier,

(a) 4 blobs

(b) 7 blobs



(c) 30 blobs



(d) 30 spaced blobs

(e) 2 circles

(f) 2 moons

Figure 1: Generated data

Sub-figure	n_b	s_b
??	4	2
??	7	2
??	30	2
??	30	6

Table 1: Parameters for the generation of the the blob shape data

they are just there to ease the visualization of them. Also, the grey color in DBSCAN means noise. Next, we explain the results of this experiment.

When $k = n_b$, EM is by far the algorithm with more accuracy when identifying the clusters, as we can see in sub-figures **??**, **??**, **??**, **??**, **??**, **??**, and in table **??**. Also, when k is incorrect, EM predicts the clusters as better as possible, as seen in sub-figure **??**. But DBSCAN is not bad. In sub-figures **??** and **??**, it predicts well the clusters, identifying the outer points as noise, which is a property that can

n_b	Algorithm	Accuracy	Sub-figure
4	EM	0.9552	??
4	DBSCAN	0.8650	??
7	EM ($k = 3$)	-	??
7	DBSCAN	0.7575	??
30	EM	0.9356	??
30	DBSCAN	0.0111	??
30 ($s_b = 6$)	EM	1	??
30 ($s_b = 6$)	DBSCAN	0.0111	??

Table 2: Accuracy of the predictions of the first experiment

be desireable in some situations. It is true that in cases with lots of small clusters it fails, as in sub-figures **??** and **??**, but the bad choice of its parameters is not discarded, although we tried several combinations of them. In these sub-figures we can also see that the separation of the clusters does not affect to the predictions, DBSCAN detects roughly the same ones in both cases. This does not happen for EM, which with more space between clusters it predicts them much better.

5.2 Second experiment

The goal of the second experiment is to observe the differences of applying EM and DBSCAN to the generated data with circle and moon shapes. For this, we used the functions `dataset.make_circles` and `dataset.make_moons` of the *scikit-learn* package. The resultant data is shown in sub-figures **??** and **??**.

Unlike the first experiment, and as sub-figures **??**, **??**, **??** and **??** show, DBSCAN predicts the clusters perfectly, while EM is unable to identify them because of their shape.

5.3 Third experiment

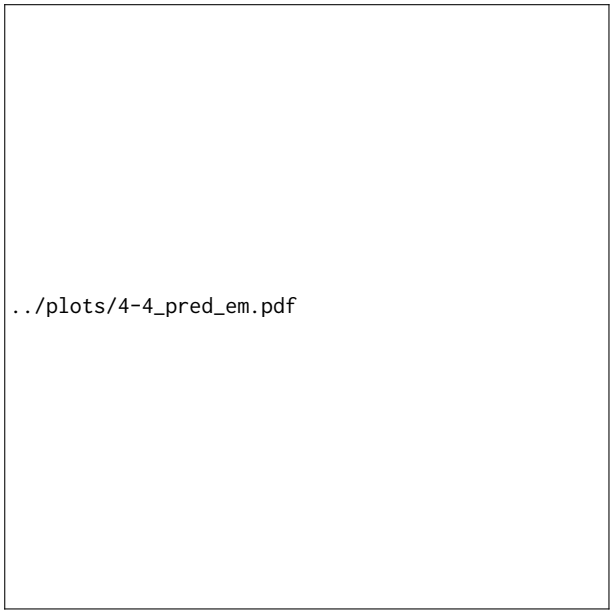
The goal of the third experiment is to observe the differences of applying EM and DBSCAN to a real dataset (Heart disease), with lots of data and dimensions, and also to compare their predictions with the results obtained in the previous work.

We tried to detect the degree of presence of heart disease in a patient using the provided data. As we did in the previous work, we made the detection first to predict if the patient has presence of hearth disease or not (binary decision), and then to predict its degree. For DBSCAN, we selected the parameters in order to obtain the desired number of clusters (with noise being a cluster itself). The parameters are $\{\varepsilon = 13, minPts = 8\}$ for the binary decision, and $\{\varepsilon = 14.7, minPts = 8\}$ for detecting the degree. For EM we selected $k = n_b$. The final results are just too bad to comment them. None of the algorithms cannot be better than selecting the clusters at random. Hence, the classifier process, developed in the previous work, is by far better. If you want to take a look at the results of the experiment, or whichever of the other two, the output of the script is in the file *'experimentation_out.txt'* in the main folder, or you can just execute it.

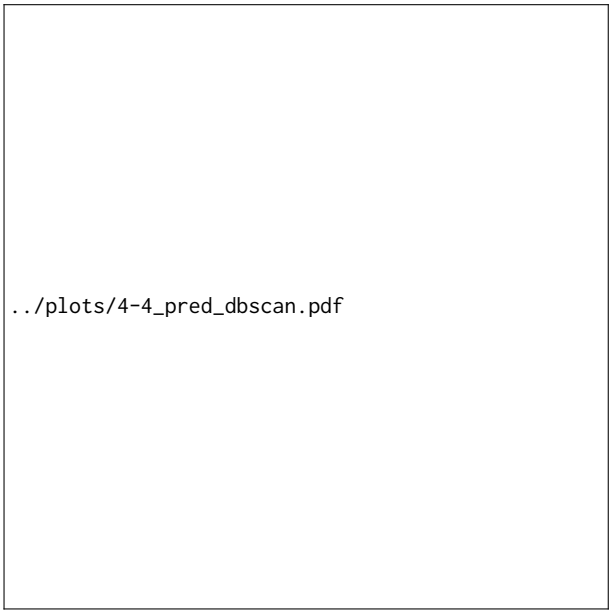
6 CONCLUSIONS

We cannot conclude this work with a clear winner. EM and DBSCAN are both good clustering algorithms, and one is better than the other

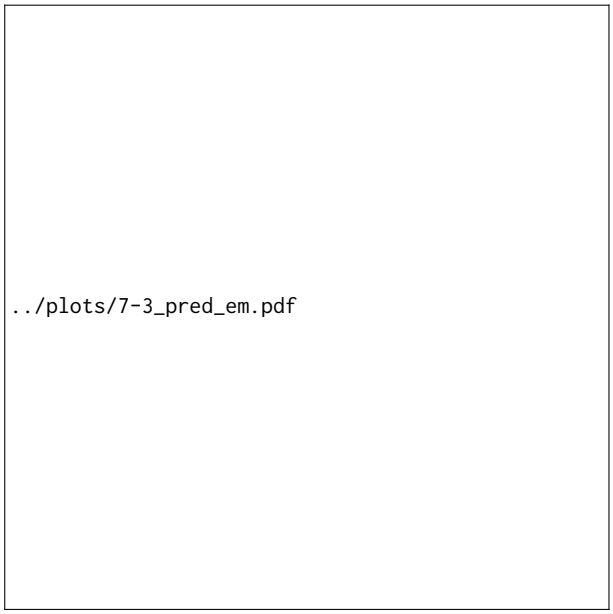
only depending on the data. As we have seen in the first and second experiment in section **??**, and with the appropriate value of k, EM predicts with high accuracy the data with blob shape, but it fails when predicting moons and circles. So, it suffers when points of the same cluster can be far one from the other, but 'connected' by the rest of the points.



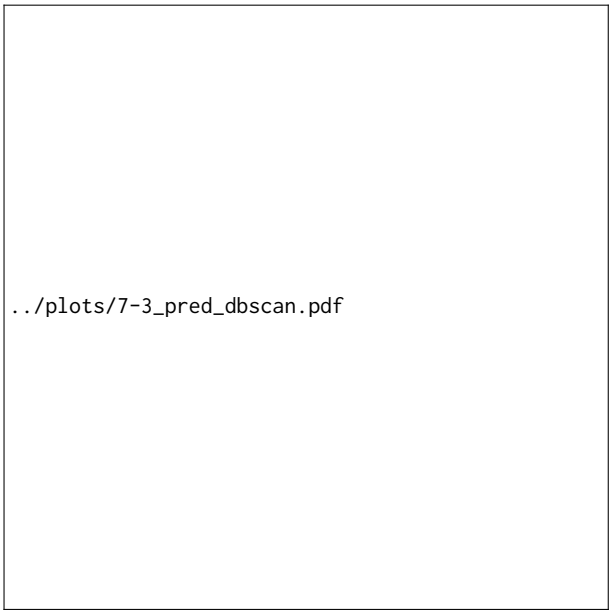
(a) EM $k = 4$



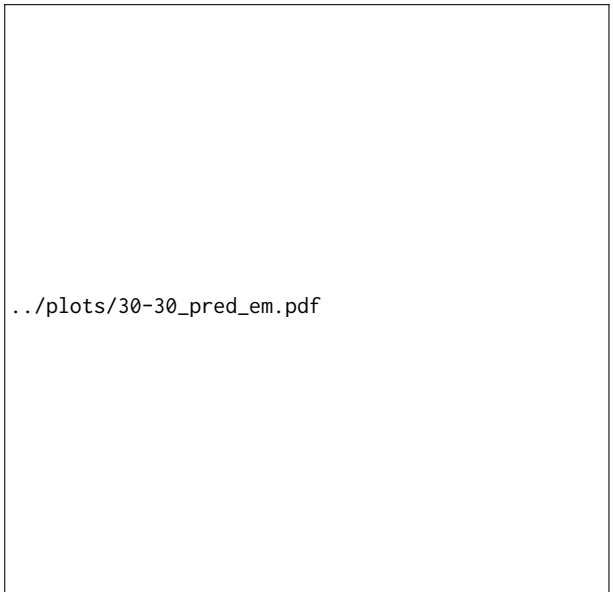
(b) DBSCAN

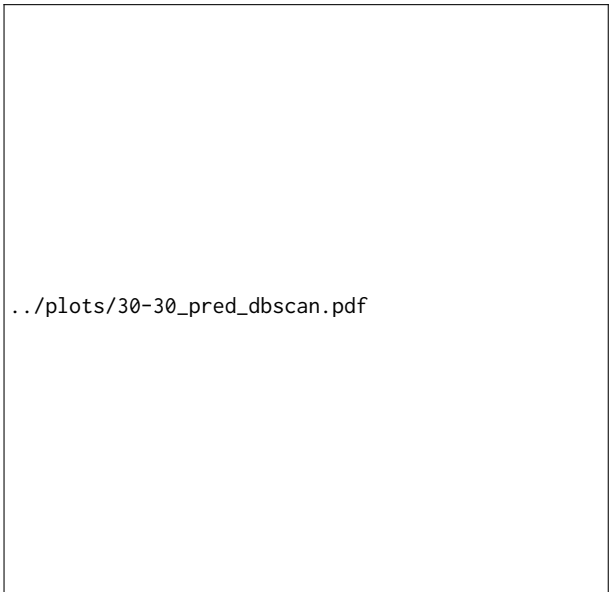


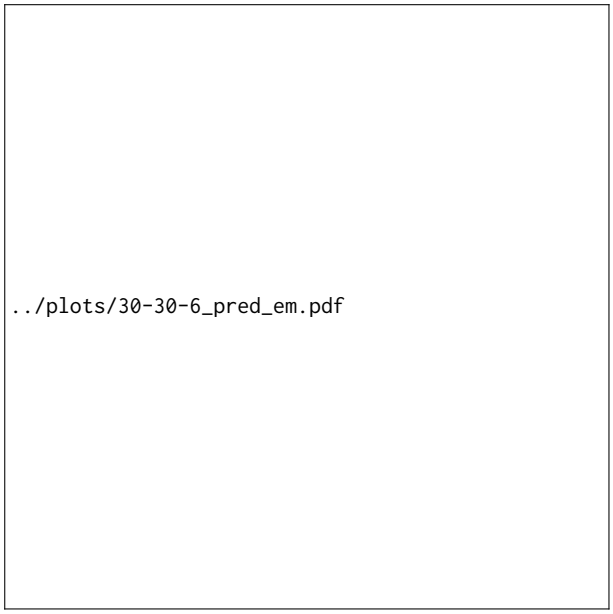
(c) EM $k = 3$



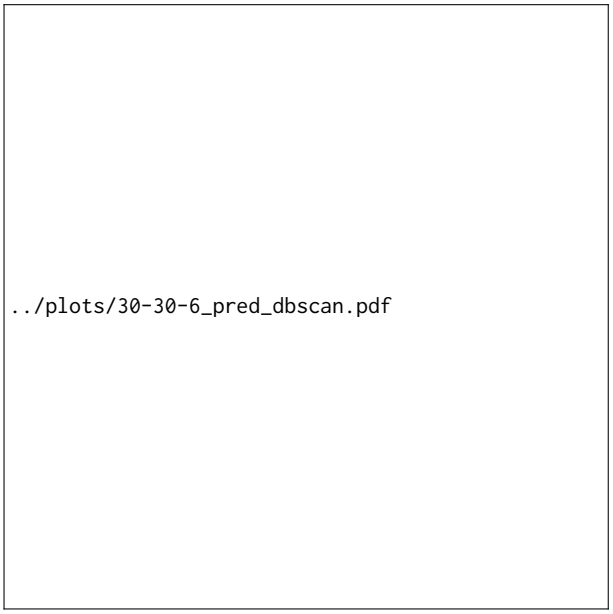
(d) DBSCAN



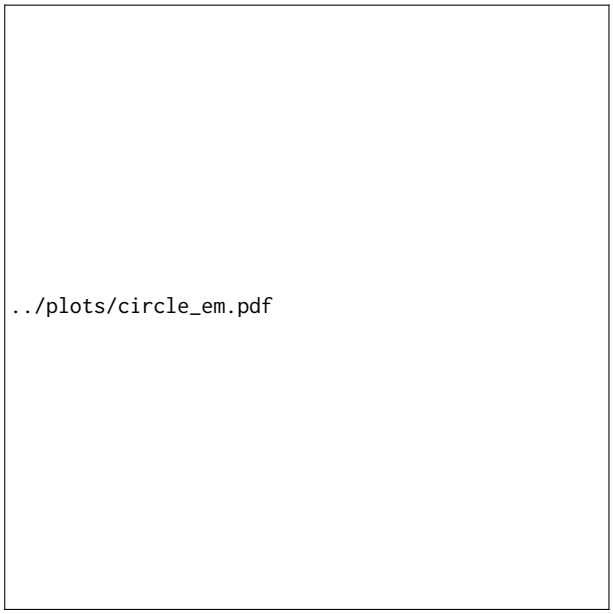




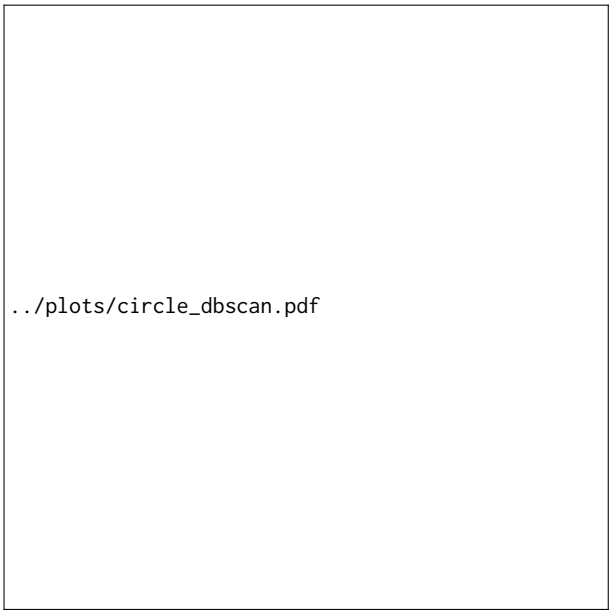
(a) EM $k = 30$



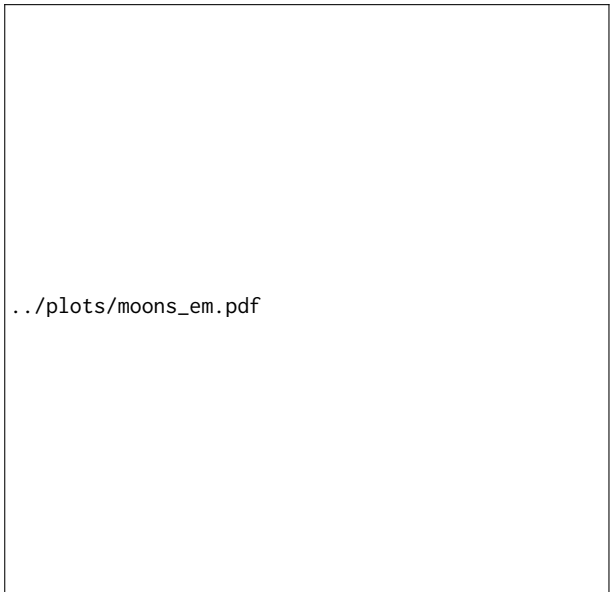
(b) DBSCAN

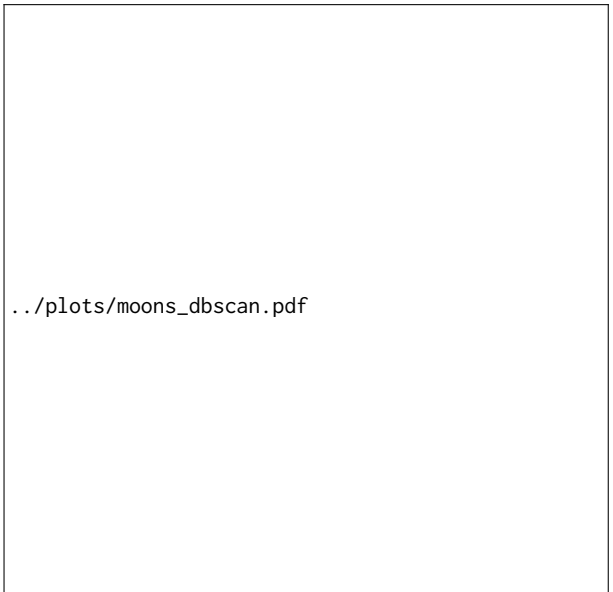


(c) EM $k = 2$



(d) DBSCAN





These could be expected because of the design of the algorithm, explained in section **??**.

On the other hand, DBSCAN, with the appropriate parameters, can also predict with high accuracy the data with blob shape, but it fails with a huge number of clusters. Unlike EM, it is also really accurate with the predictions of moons and circles, since it uses the previous mentioned 'connection' of the points in order to decide the clusters. Furthermore, the prediction of noise can be really useful with some data.

In both cases, the prediction of the real dataset was horrible. There are problems that the clustering algorithms are just not made for them, and the Heart disease dataset is one of those. Better use the classification algorithms in this cases.