EM v.s. DBSCAN

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

ABSTRACT

1 INTRODUCTION

2 ALGORITHMS

2.1 EM

Expectation-maximization algorithm (EM) [1, 5, 7] 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) [2, 6] 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.

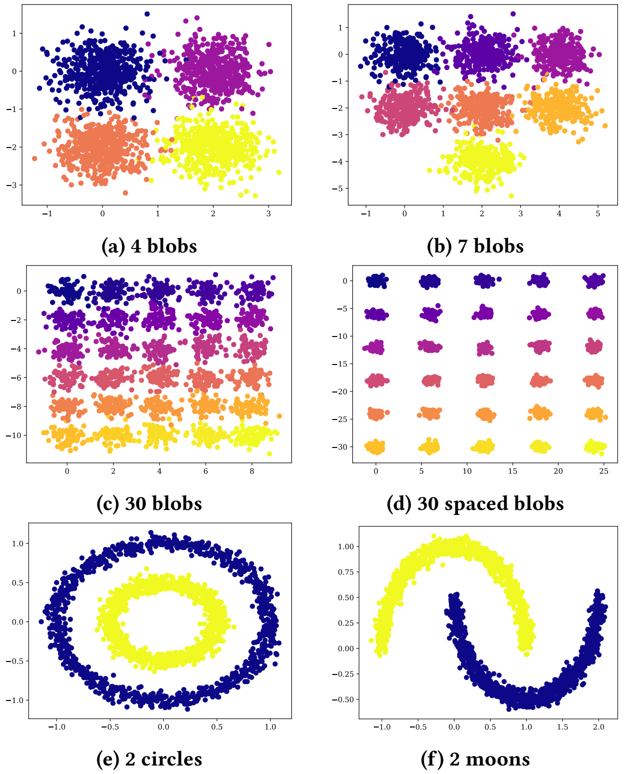
Given ε and *minPts*, the DBSCAN algorithm can be abstracted into the following steps:

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* [4] 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 1.

We also used the dataset of the previous work, which is the Heart Disease dataset [3], to see how the algorithms behave with realistic data.



(a) 4 blobs

(b) 7 blobs

(c) 30 blobs

(d) 30 spaced blobs

(e) 2 circles

(f) 2 moons

Figure 1: Generated data

4 EXPERIMENTATION

4.1 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 1). 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.

Sub-figure	n_b	s_b
1a	4	2
1b	7	2
1c	30	2
1d	30	6

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

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. Next, we explain the results of this experiment.

- **4 blobs.**

5 IMPLEMENTATION

6 CONCLUSIONS

REFERENCES

[1] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological) 39*, 1 (1977), 1–38.

[2] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press, pp. 226–231.

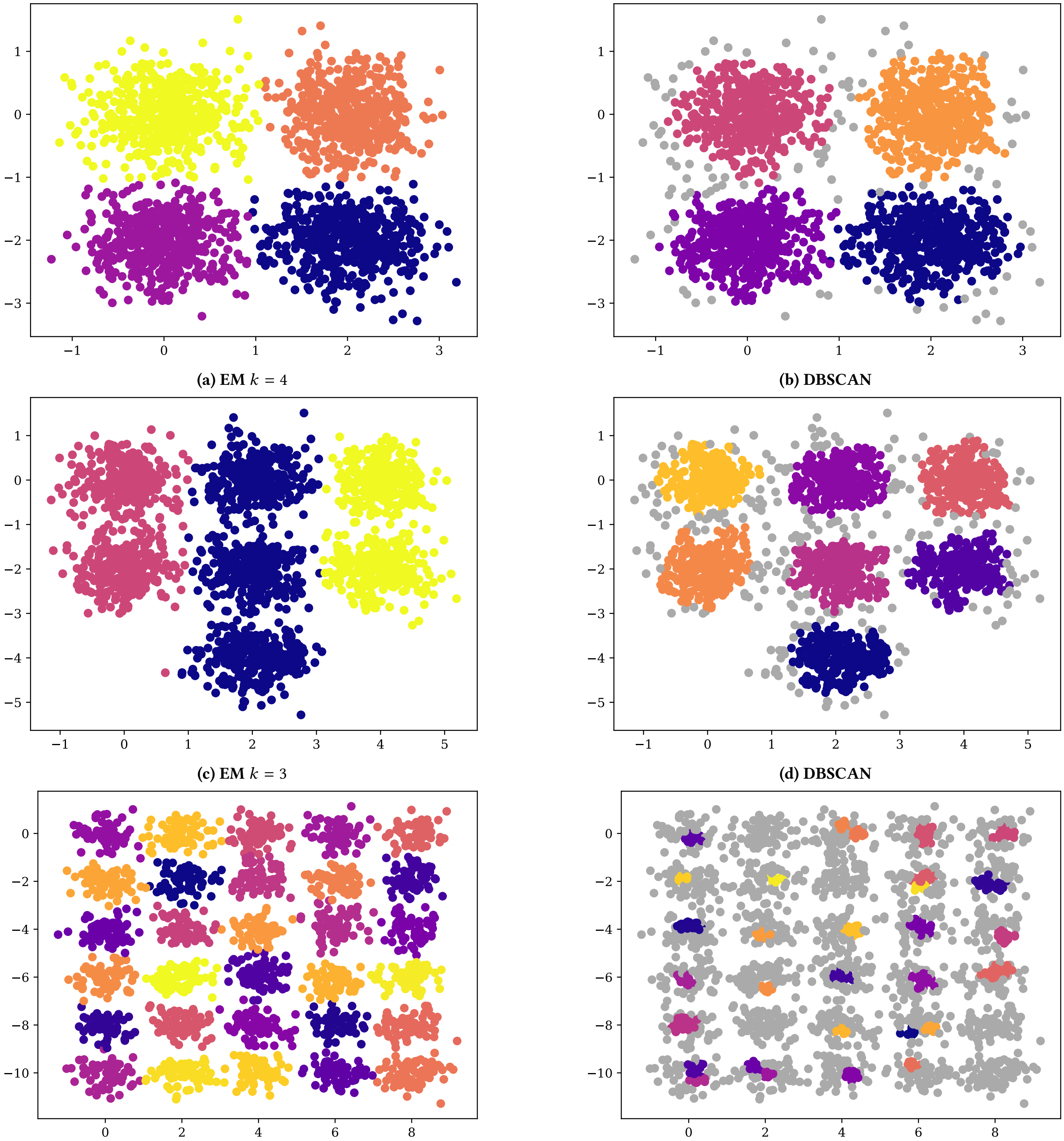
[3] Janosi, A., Steinbrunn, W., Pfisterer, M., and Detrano, R. Heart disease. https://archive.ics.uci.edu/ml/datasets/heart+Disease, 2019.

[4] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[5] Rapidminer. Expectation maximization clustering. https://docs.rapidminer.com/latest/studio/operators/modeling/segmentation/expectation_maximization_clustering.html, 2019.

[6] Wikipedia. Density-based spatial clustering of applications with noise. https://en.wikipedia.org/wiki/DBSCAN, 2019.

[7] Wikipedia. Expectation–maximization algorithm. https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm, 2019.



(a) EM $k = 4$

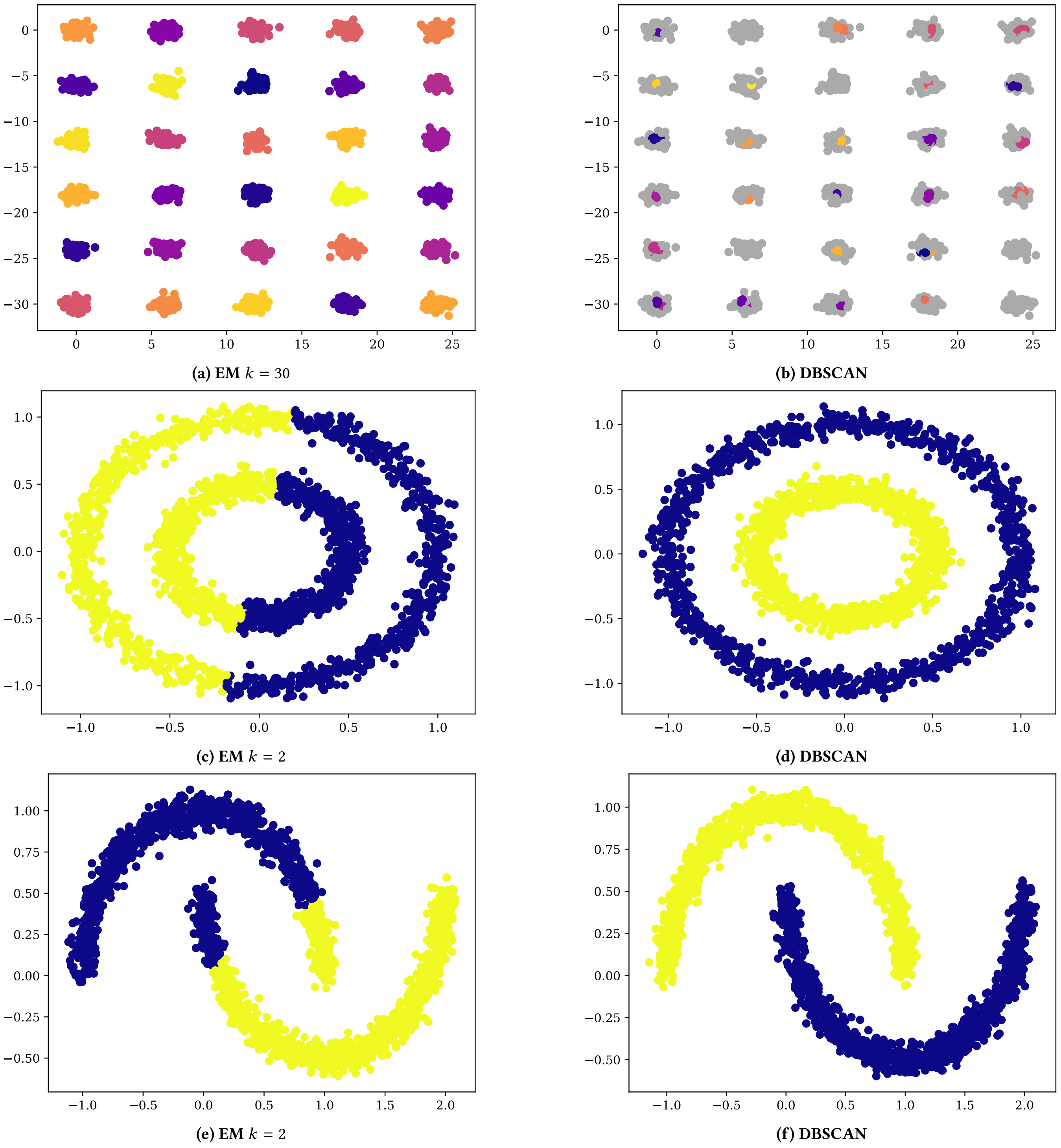
(b) DBSCAN

(c) EM $k = 3$

(d) DBSCAN

(e) EM $k = 30$

(f) DBSCAN



(a) EM $k = 30$

(b) DBSCAN

(c) EM $k = 2$

(d) DBSCAN

(e) EM $k = 2$

(f) DBSCAN