

## Bayesian Gaussian Mixture Models

Rather than manually searching for the optimal number of clusters, you can use the `BayesianGaussianMixture` class, which is capable of giving weights equal (or close) to zero to unnecessary clusters. Set the number of clusters `n_components` to a value that you have good reason to believe is greater than the optimal number of clusters (this assumes some minimal knowledge about the problem at hand), and the algorithm will eliminate the unnecessary clusters automatically. For example, let's set the number of clusters to 10 and see what happens:

```
>>> from sklearn.mixture import BayesianGaussianMixture
>>> bgm = BayesianGaussianMixture(n_components=10, n_init=10,
>>> random_state=42)
>>> bgm.fit(X)
>>> bgm.weights_.round(2)
array([0.4 , 0.21, 0.4 , 0. , 0. , 0. , 0. , 0. , 0. , 0. ])
```

Perfect: the algorithm automatically detected that only three clusters are needed, and the resulting clusters are almost identical to the ones in [Figure 9-16](#).

A final note about Gaussian mixture models: although they work great on clusters with ellipsoidal shapes, they don't do so well with clusters of very different shapes. For example, let's see what happens if we use a Bayesian Gaussian mixture model to cluster the moons dataset (see [Figure 9-21](#)).

Oops! The algorithm desperately searched for ellipsoids, so it found eight different clusters instead of two. The density estimation is not too bad, so this model could perhaps be used for anomaly detection, but it failed to identify the two moons. To conclude this chapter, let's take a quick look at a few algorithms capable of dealing with arbitrarily shaped clusters.



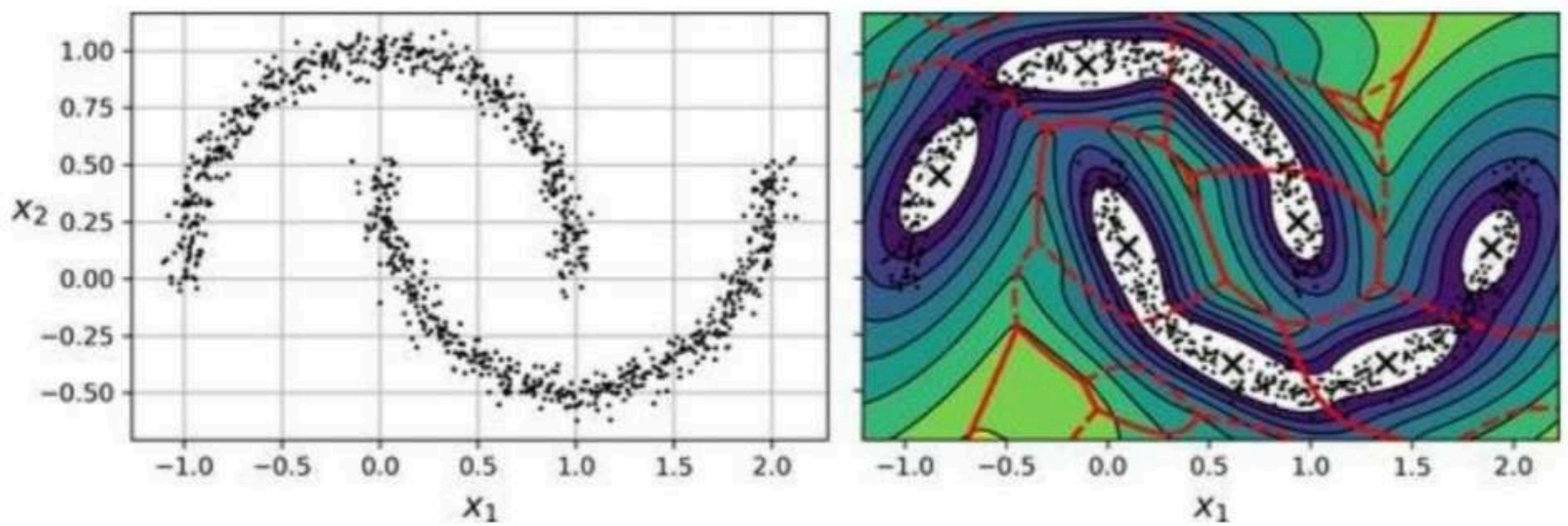


Figure 9-21. Fitting a Gaussian mixture to nonellipsoidal clusters

## Other Algorithms for Anomaly and Novelty Detection

Scikit-Learn implements other algorithms dedicated to anomaly detection or novelty detection:

*Fast-MCD (minimum covariance determinant)*

Implemented by the `EllipticEnvelope` class, this algorithm is useful for outlier detection, in particular to clean up a dataset. It assumes that the normal instances (inliers) are generated from a single Gaussian distribution (not a mixture). It also assumes that the dataset is contaminated with outliers that were not generated from this Gaussian distribution. When the algorithm estimates the parameters of the Gaussian distribution (i.e., the shape of the elliptic envelope around the inliers), it is careful to ignore the instances that are most likely outliers. This technique gives a better estimation of the elliptic envelope and thus makes the algorithm better at identifying the outliers.

*Isolation forest*

This is an efficient algorithm for outlier detection, especially in high-dimensional datasets. The algorithm builds a random forest in which each decision tree is grown randomly: at each node, it picks a feature randomly, then it picks a random threshold value (between the min and max values) to split the dataset in two. The dataset gradually gets chopped into pieces this way, until all instances end up isolated from the other instances. Anomalies are usually far from other instances, so on average (across all the decision trees) they tend to get isolated in fewer steps than normal instances.



## *Local outlier factor (LOF)*

This algorithm is also good for outlier detection. It compares the density of instances around a given instance to the density around its neighbors. An anomaly is often more isolated than its  $k$ -nearest neighbors.

## *One-class SVM*

This algorithm is better suited for novelty detection. Recall that a kernelized SVM classifier separates two classes by first (implicitly) mapping all the instances to a high-dimensional space, then separating the two classes using a linear SVM classifier within this high-dimensional space (see [Chapter 5](#)). Since we just have one class of instances, the one-class SVM algorithm instead tries to separate the instances in high-dimensional space from the origin. In the original space, this will correspond to finding a small region that encompasses all the instances. If a new instance does not fall within this region, it is an anomaly. There are a few hyperparameters to tweak: the usual ones for a kernelized SVM, plus a margin hyperparameter that corresponds to the probability of a new instance being mistakenly considered as novel when it is in fact normal. It works great, especially with high-dimensional datasets, but like all SVMs it does not scale to large datasets.

## *PCA and other dimensionality reduction techniques with an `inverse_transform()` method*

If you compare the reconstruction error of a normal instance with the reconstruction error of an anomaly, the latter will usually be much larger. This is a simple and often quite efficient anomaly detection approach (see this chapter's exercises for an example).

## **Exercises**

1. How would you define clustering? Can you name a few clustering algorithms?
2. What are some of the main applications of clustering algorithms?
3. Describe two techniques to select the right number of clusters when using