Anomaly detection in a toy 2D dataset

Yolan Uyttenhove

April 7, 2019

1 Inspecting the dataset

A scatter plot of the toy dataset is shown in figure 1. On visual inspection, the data seems to be structured into three to five soft clusters. My algorithm for anomaly detection on this dataset exploits this apparent structure in the dataset. The downside of this is that the algorithm becomes tailor made for this and similar datasets and will not perform as well for differently structured datasets (for example datasets with a strong serial structure). The benefit is that it will potentially perform better on this and similar datasets than a more general algorithm would.

2 The algorithm

In a nutshell, the algorithm I designed fits a number of weighted Gaussian distributions to the data and flags data points with a likelyhood below a certain threshold (significance level) as outliers.

2.1 Determining the optimal number of Gaussian distributions

I employed two methods to determine the optimal number, $1 \leqslant N \leqslant 9$, of weighted Gaussian distributions to model the data. For the first method I followed an 8-fold cross validation procedure. In each fold, a mixture of N independent, weighted Gaussian distributions were fitted to the train split (87.5%) of the data, after which the average log-likelyhood $l_{\text{fold}_i}(N)$ of the samples from the test split (12.5%) of the data for that combination of Gaussian distributions was calculated. Finally the average of the log-likelyhood over the different folds is calculated as follows:

$$l(N) = \sum_{i=1}^{8} l_{\text{fold}_i}(N)/8.$$

The objective now is two-fold. We want to maximise $l(N)$ (maximise the likelyhood of the model) while minimising N (the amount of Gaussian distributions and thus proportional to the number of parameters of the model) to avoid overfitting. The results of this procedure are plotted in figure 2. For $N \geqslant 5$, the increase in log-likelyhood really diminishes. The optimal number of Gaussian distributions could be chosen as 4 or 5.

The second procedure uses the Akaike information criterion (AIC) and the Bayesian information criterion (BIC) (which are closely related) to aid in decision making. These are defined as follows: AIC $= k - 2l$ and BIC $= k\ln(n) - 2l$, with $k \propto N$ the amount of parameters of the model, l the log-likelyhood of the data under the model and n the number of data points. The model with the lowest information criteria is preferred (minimising the information criteria comes down to maximising the log-likelyhood with an extra penalty term for the number of parameters of the model). For each N value, the averages of the AIC and BIC over 100 bootstrap resamplings (with replacement) from the original dataset were calculated. The bootstrap resampling was used to minimise the influence of outliers on the (mean) values of the information criteria. The results are displayed in figure 3. The AIC shows a strong decrease up to $N = 5$, after which it decreases much more

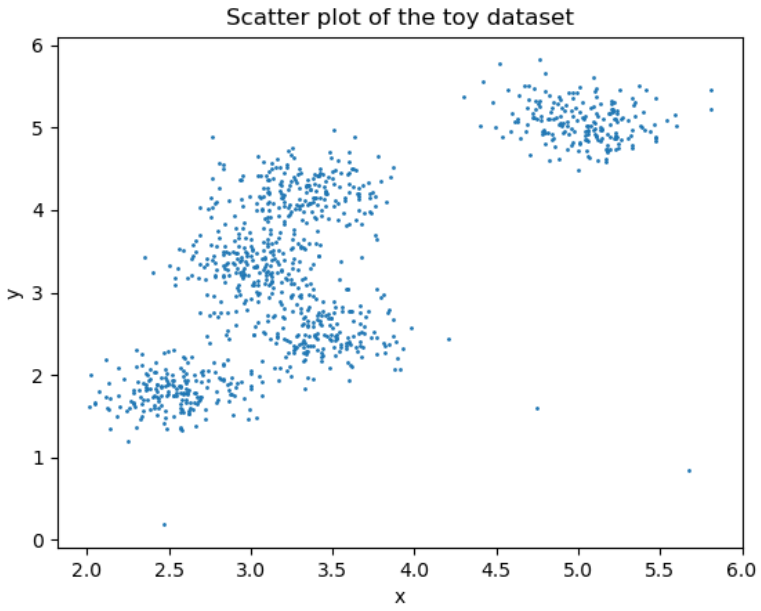

Figure 1: The data seems to be structured into 3 to 5 (soft) clusters

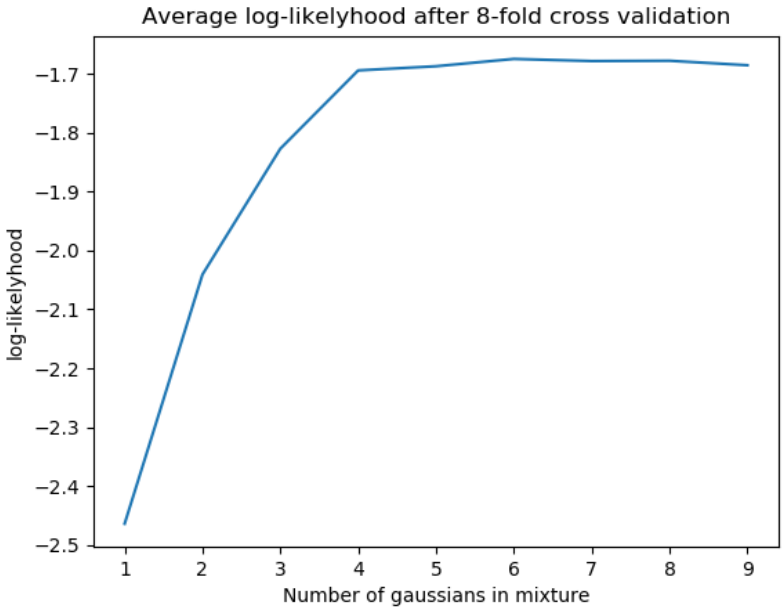

Figure 2: For $N \geqslant 5$ the increase in log-likelyhood becomes negligible, if any.

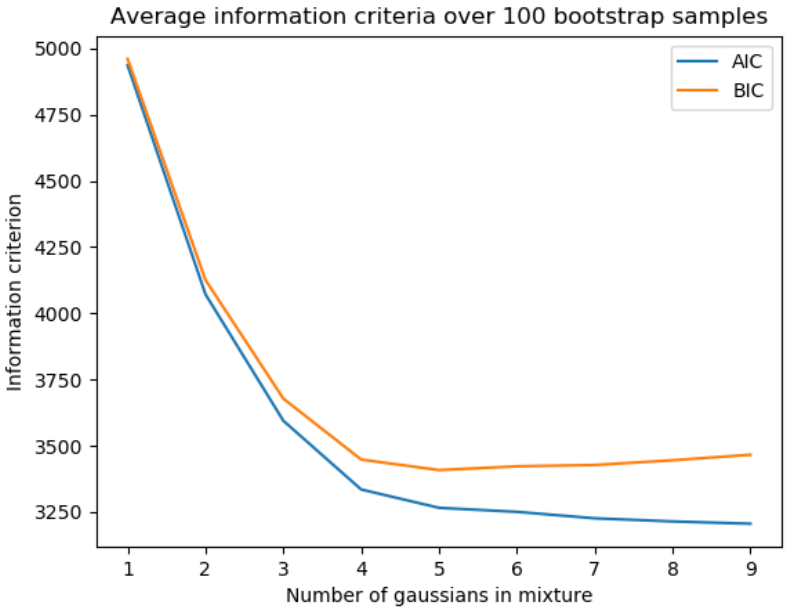

Figure 3: Again a mixture of 5 Gaussian distributions seems optimal.

slowly and the BIC shows a minimum for $N = 5$. So a mixture of 5 Gaussian distributions seems optimal, which is in line with the results from the first method.

2.2 An iterative Gaussian mixture model (GMM) algorithm to detect anomalies

Like k-means clustering, although much less severely, GMM's are sensitive to the presence of outliers in the dataset on which they are fitted. I try to minimise the influence of outliers by taking an iterative approach. An initial model is fitted, which is then used to discard the datapoints with likelyhood below a certain threshold (0.01) from the dataset and a new model is fitted on the remaining datapoints. This procedure is repeated until no more datapoints are removed. The final model obtained in this way is then used to flag the points from the original dataset with likelyhood (for the final model) below 0.01 as outliers. The GMM fitting is done via `sklearn.mixture.GaussianMixture`, which uses expectation-maximisation to obtain a (locally) optimal fit. The result of this analysis is shown in figure 4, together with a contour plot of the Gaussian distributions of the final model.

2.3 Comparing my algorithm to DBSCAN

To assess the performance of my algorithm I also implemented a simple (and fairly unoptimized) version of the DBSCAN algorithm to detect outliers. Without going into too much detail about the DBSCAN algorithm, I will quickly justify my choice of parameters for the algorithm. If we plot the distance for every point to it's N nearest neighbours in ascending order (see figure 6) we see a strong turn-up in distance for the ± 20 most *isolated* points, indicating that these are probably outliers. and this number is weakly dependent on N (as can be seen in figure 6). So a good choice of parameters are $N = 5$ and eps $= 0.2462$, the largest distance to the 5 nearest neighbours of points before the *elbow point* in fig. 6, or equivalently the smallest distance to the five nearest neighbours for the 20 most isolated points. The results of the DBSCAN algorithm are shown in figure 5. Both algorithms give very similar results, with the main difference being that the GMM algorithm flags a small amount of extra points as outliers, but looking at the graphs this seems completely reasonable. The advantages of the DBSCAN algorithm are that it is model agnostic and requires only a distance measure

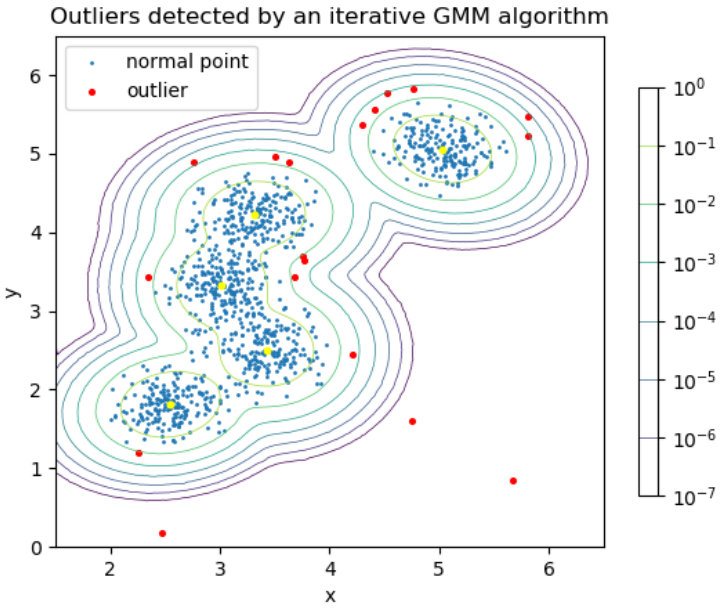

Figure 4: The outliers detected with my algorithm (using a threshold of 0.01).

and should thus be more generally applicable. The advantages of my algorithm on the other hand are that it is more tailored towards this type of data and perhaps more importantly: once a model has been fitted, one can trivially score new datapoints based on that model, whereas the DBSCAN algorithm needs to be run again on the whole dataset with the extra data point. Also rather than giving a discrete answer (*yes* or *no*) as the DBSCAN algorithm would, the GMM algorithm returns a score (the likelyhood) based on which a more nuanced decision can be taken.

3 REST API

I implemented a simple REST API (using the `flask` and `flask_restful` modules) that supports the following requests:

- `get`: returns the previously submitted data point, if any.
- `put`: submit a new data point as json message and returns the score of this data point for the model trained on the training dataset.
- `delete`: delete the previously submitted data point, if any.

If speed is a priority the model can be computed once from the training data, after which submitted data points are scored against this model and the scores are returned. This is the current implementation. However, this behaviour can be trivially changed to retrain the model every N submitted data points, adding these submitted datapoints to the training dataset, or even retrain the model for every submitted data point if speed is not a high priority.

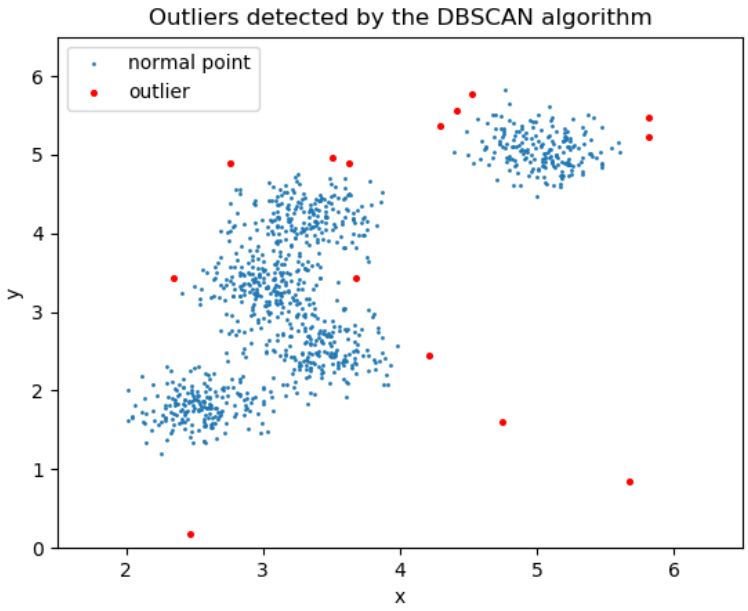

Figure 5: The outliers detected by the DBSCAN algorithm

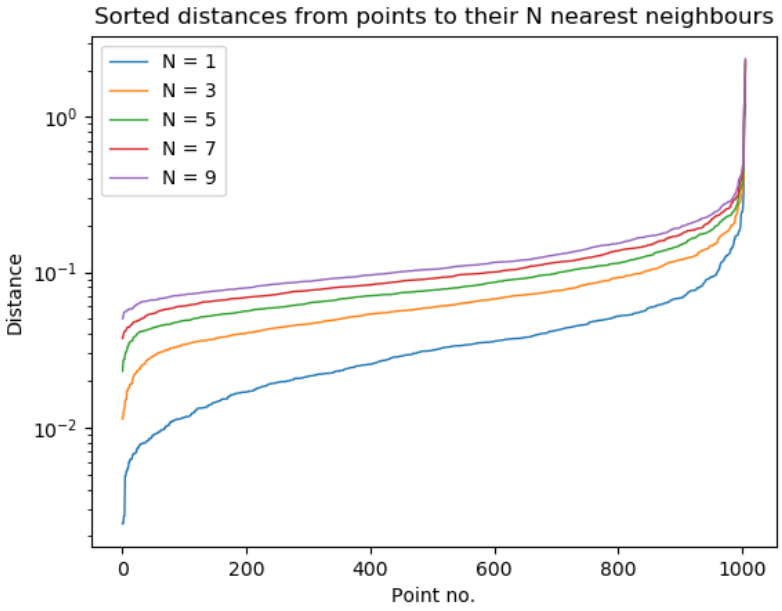

Figure 6: There is a strong turn-up in distance to the N nearest neighbours for the ± 20 most isolated points.