Foundations of Machine Learning - Lab 6

Johnny Joyce (jvjj1u19@soton.ac.uk)

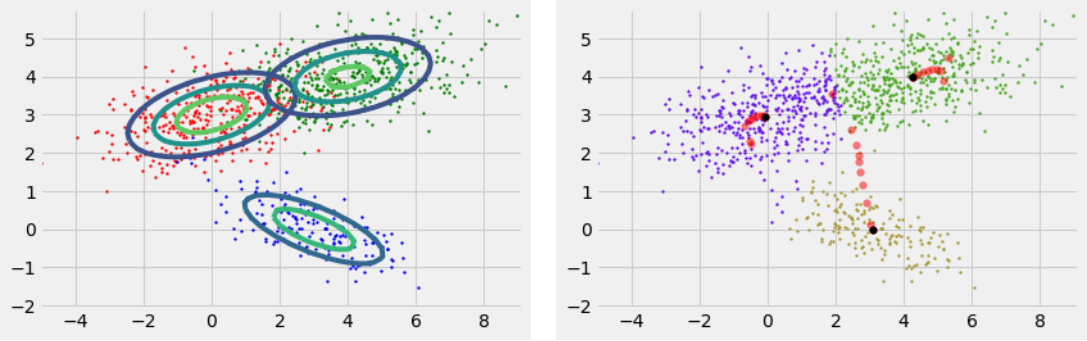
December 6, 2019

The previous machine learning labs have been completed and the feedback from labs 1-3 have been taken into account. Feedback on lab 3 asked how the ROC curve for the Mahalanobis classifier was calculated. This was done by setting a threshold ranging from the minimum of the Mahalanobis distances of our data to the maximum of Mahalanobis distances. The threshold was varied over this interval as the threshold for classification in either class and the percentage of true positives and false positives was plotted. Additionally, the lab 3 report contained an ROC curve below the 45° line. This situation is always avoidable as one can simply reverse the classifier to consider positives as negatives and vice versa.

1 K-Means Clustering

1.1 Implementing K-means

The given code was used to create three bivariate Gaussian distributions. Figure 1 displays the results of the self-implemented K-means algorithm. We can see from Figure 1(b) that the algorithm correctly places most coordinates in a cluster corresponding to their respective Gaussian distributions, though there are some near the boundaries where two clusters meet that are not correctly classified. This is to be expected, since the K-means algorithm always produces a Voronoi diagram, which will not always perfectly approximate bivariate Gaussian data.



(a) (b)

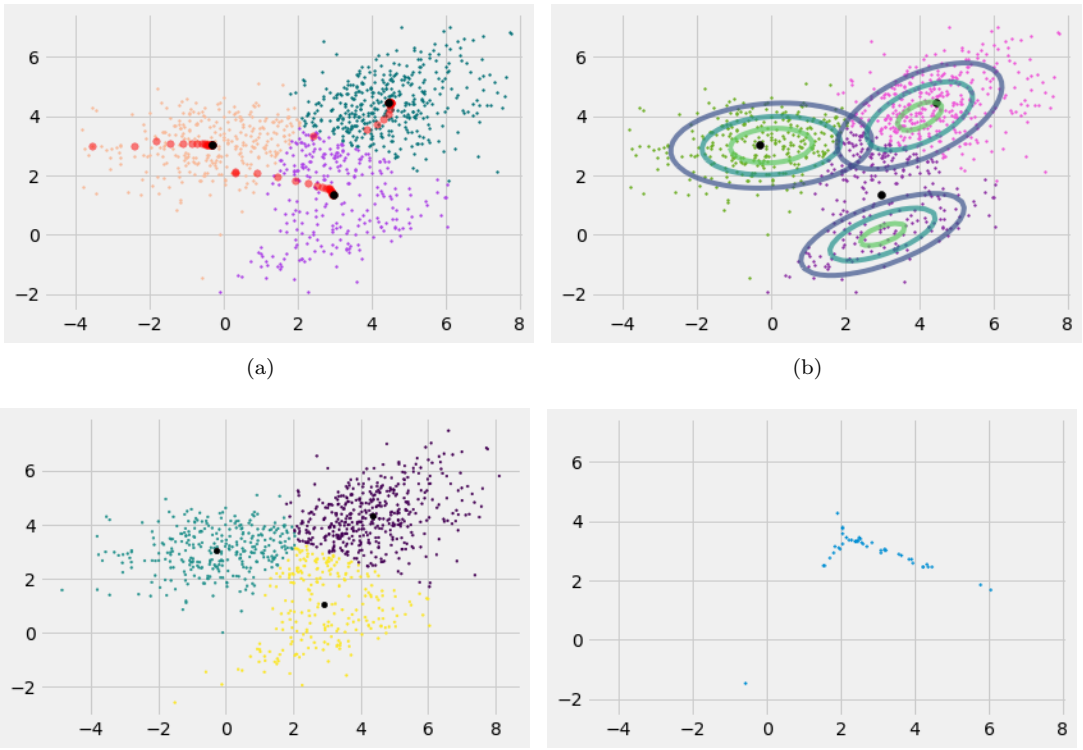
Figure 1: (a) Plot of three bivariate Gaussian distributions and their respective contours (b) The results of K-means clustering on said distributions. Black coordinates indicate cluster centres, red coordinates indicate previous cluster centres, and all other colours indicate clusters.

We can find the accuracy of our algorithm by finding the permutation of assignments of clusters to distributions that maximises the number of correctly classified elements. In this case, 91% of coordinates were correctly classified.

1.2 Comparing with contours and comparing with sklearn

Let us now compare the performance of our implemented K-means algorithm to the version included in the skikit-learn (sklearn) package. We will use data sampled from distributions whose contours are shown in Figure 2(b). Figure 2(b) also displays the performance of our algorithm with the contours overlaid above the diagram, and we can see that it performs relatively well through visuals alone.

Figure 2(a) shows the performance of our algorithm without the contours and Figure 2(c) shows the performance of the sklearn implementation. Both Figures show fairly similar results. However, by examining Figure 2(d), we see that there are some of the coordinates on the boundary lines of two clusters were assigned different clusters. The sklearn implementation scored better than the self-implemented version, correctly classifying 83% coordinates compared to the self-implementation's 80%.



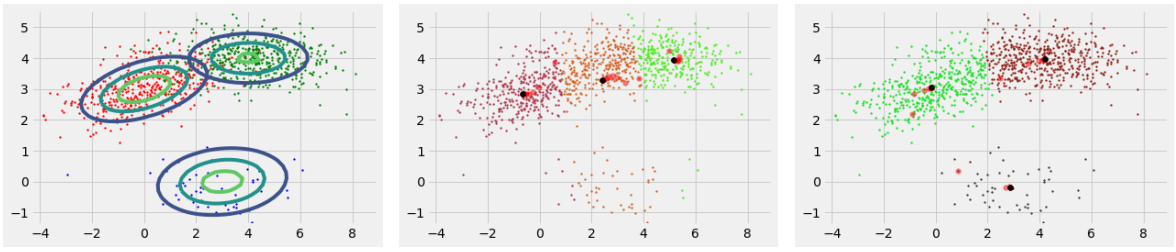
(a) (b) (c) (d)

Figure 2: (a) The results of running the self-implemented K-means algorithm on a new set of bivariate Gaussian distributions (b) The results of the self-implemented K-means with the contours of the Gaussians (c) The results of sklearn's K-means algorithm (d) The coordinates where sent to different clusters by the self-implemented K-means compared to the sklearn's K-means.

1.3 Dependence on initial guesses

We can now investigate whether the performance of K-means algorithm depends on the initial guesses for cluster centres. To do so, let us use the distributions in Figure 3(a). We can see visually that the algorithm performed significantly worse in Figure 3(b) (with 652 of 998 correctly classified) than in Figure 3(c) (906 of 998 correctly classified).

In Figure 3(b), the centre of the middle cluster moved to the left of its original position instead of downwards. This may be because the coordinates in the distribution near the horizontal axis had relatively few samples, meaning that they would have little effect on the recalculation of distant cluster centres.



(a) (b) (c)

Figure 3: (a) A new selection of Gaussian distributions and their contours (b)(c) The results of the self-implemented K-means algorithm with different starting guesses on the new distributions

1.4 Testing on an external dataset

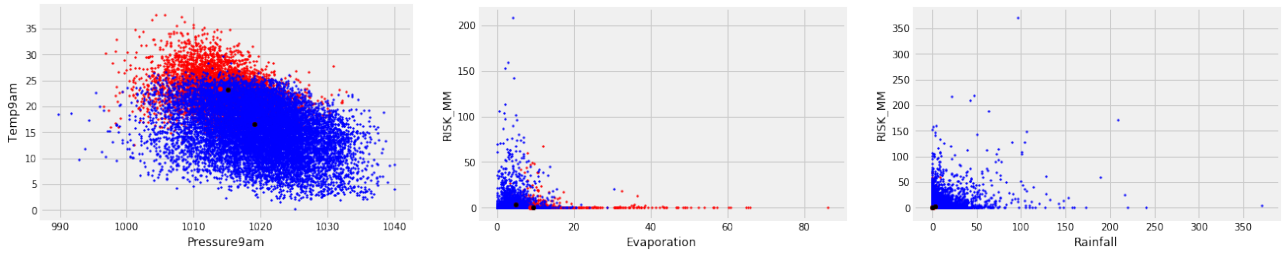
Let us now test our model on a new dataset. We will be using the "Rain in Australia" dataset found at `https://www.kaggle.com/jsphyg/weather-dataset-rattle-package/`, which contains weather data in Australia and

a binary classification as to whether it will rain the following day or not. For practicality, the data was restricted to the first 20,000 entries. Non-numerical attributes were excluded.

Initial tests performed very poorly, only correctly classifying around 50% of the data; this is the bare minimum performance that a binary classifier could achieve, as if one were to correctly classify 40% of data for example, the two classifications could be swapped to obtain 60% performance.

The model was updated to account for data with missing entries in certain attributes. When calculating means and distances, the algorithm would not include NoneType entries in calculations. Functionality was also added to normalise each attribute to have a mean of 0 and a standard deviation of 1. For example, the "Rain in Australia" dataset has an attribute "Humidity at 3PM" with a standard deviation of 21.3 and "Number of hours of bright sunshine" with a deviation of 3.8. By normalising, the variances within each attribute will be considered equally important to one another so each attribute will have the same effect on the calculation of cluster centres.

Figures 4(a) - 4(c) show how the classifier classed entries with two of their attributes plotted.



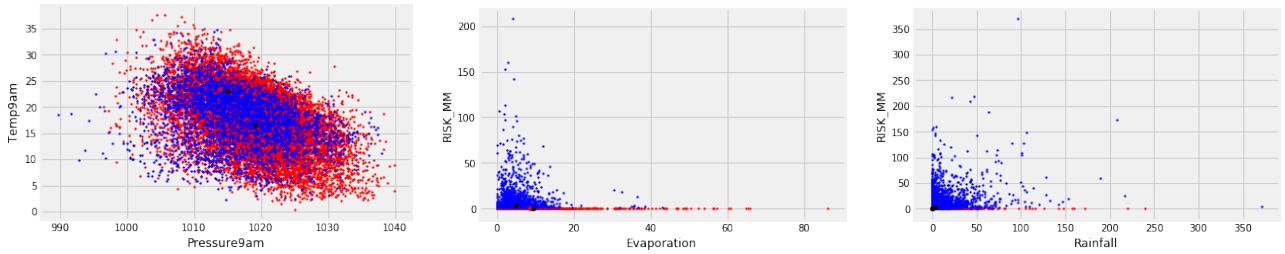
(a) x: Pressure at 9AM (hPa). y: Temperature at 3PM (°C).

(b) x: 24h Evaporation before 9AM (mm). y: Risk of rain for tomorrow.

(c) x: The rainfall for the day (mm). y: The risk of rain for tomorrow.

Figure 4: Predictions for made by the improved K-means algorithm as to whether it will rain tomorrow or not. Blue indicates predicted rain, red indicates no predicted rain.

Figure 4(b) appears to show that the classifier is making reasonable conclusions based on the evaporation in the past 24 hours and the risk of rain. When the given predictor predicts a high risk of rain, the classifier also does so. Similarly, if there has not been much evaporation in the past 24 hours, then it predicts that it will not rain tomorrow (possibly because low evaporation may imply that there are few rain clouds). Similarly, Figure 4(c) also seems to show reasonable conclusions; the days where rain is not predicted are all tightly packed around the origin, where both rainfall for the current day and the risk of rain are low. However, Figure 5 shows the true value of whether it rained or not the following day, and we can see that Figure 4(a) was fairly inaccurate, and also that Figure4(c) produced many false positives for days when there was zero risk of rain.



(a) x: Pressure at 9AM (hPa). y: Temperature at 3PM (°C).

(b) x: 24h Evaporation before 9AM (mm). y: Risk of rain for tomorrow.

(c) x: The rainfall for the day (mm). y: The risk of rain for tomorrow.

Figure 5: The *true* values of whether it rained the following day. (a),(b),(c) correspond to their respective Subfigures in Figure 4.

The classifier achieved an F1-score of 0.72 and only correctly classified 57% of the data it was given. This may be because clustering is an unsuitable approach to this particular problem and that another approach such as a Bayesian classifier would be more effective. It may also be because the non-numerical attributes were deleted, leaving 17 of the original 23 attributes, as well as the fact that out of every attribute for every data point, 18% of the data was missing.