Individual Assignment TDT4173

I chose the algorithms k-means clustering and logistic regression.

K-means clustering works by randomly choosing k points as centroids and then assigning surrounding datapoints to their closest centroid. This allows the centroids to be moved to the average of its assigned data. Then these steps are repeated until a criterion is met or there are no more changes happening. The algorithm assumes that data from the same class appear in the same neighborhood. This makes k-means work well for problems where data from each class is close together and preferably in circular groups. It does not work well when the classes appear in different shapes, like lines or curves.

A standard algorithm using Andrew Ng's lecture notes worked fine for the data in task 1 but poorly for task 2. To make the algorithm work for both tasks I added data normalization to the range [0, 1] and set up the algorithm to train the centroids n = 100 times finding the centroids giving the lowest distortion and saving those for use in prediction. This made the algorithm a lot less reliant on the initial centroids as it got a lot of chances to find centroids leading to an acceptable minimum.

Result task 1:

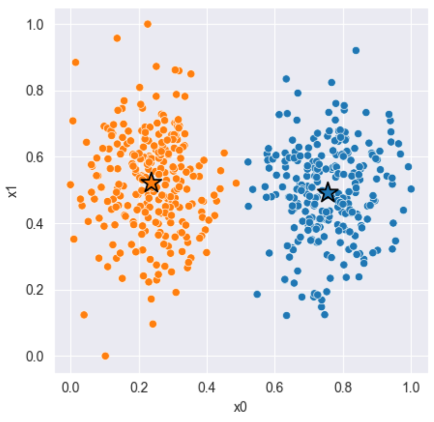

Figure 1 We can see that the centroids are properly placed and groups the data well.

Result task 2:

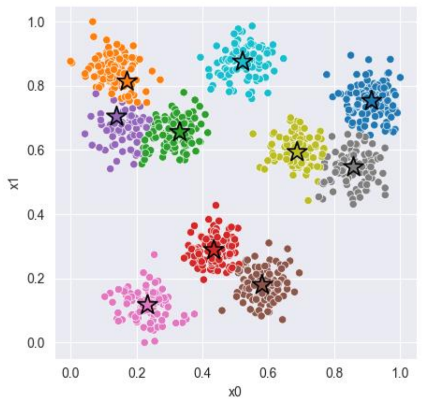

Figure 2 Again we see that the algorithm found all the clusters. However the centroids are less centered here

In the second dataset there are more classes, and many of the classes are close to each other. This often results in two centroids being assigned to one class, leaving one other class without a centroid. There is also a difference in the scale along each feature, making the problem suited for normalization. My modifications were as mentioned normalizing the data and letting the algorithm run several times to find the best classification out of all the tries.

Logistic regression uses a sigmoid function to classify a binary target by assuming there exists a boundary that splits 0s from 1s. This boundary is found by using gradient ascent to maximize the probability likelihood of classifying data correctly. This works well for problems with a linear decision boundary, while for nonlinear boundaries or in cases with a lot of outliers logistic regression will perform worse unless the data can be transformed to a more fitting space.

For logistic regression I made the algorithm following Andrew Ng's lecture notes on supervised learning. This performed well on the first dataset but really poorly on the second. To classify the second dataset i added normalization to zero mean and unit variance and transferred the data into polar coordinates as the data was following a circular pattern which looked really fitting for polar coordinates.

Result task 1:

Accuracy: 0.940
Cross Entropy: 0.177

Result task 2:

Train

Accuracy: 0.902

Cross Entropy: 0.226

Test

Accuracy: 0.908

Cross Entropy: 0.209

For both tasks the accuracy was over 90% and equally good for train and test data. The second problem has data with a nonlinear boundary between the two classes. I solved this problem by transforming the data into polar coordinates, making the boundary linear, which resulted in the simple logistic regression algorithm being able to properly classify the data.