Solutions to seminar Week 4: Clustering

The aim of this seminar was to give you practical experience with the `K-means` clustering algorithm. The key point is that the choice of k is critical. When prior knowledge exists about the number of classes (e.g., you are told in advance that the data you are classifying consists of items of just n classes OR you have worked it out, perhaps through probability density estimation) then the method can prove quite powerful although it is easy to trip it (see Question 3). If, on the other hand, no prior knowledge is available, then you need to treat k as a hyper-parameter, i.e., a model parameter that needs to be optimised. As always, your guiding principle is whether a choice of k improves or not generalisation.

Question 1

Here, many of you got sidetracked by my use of the word 'unimodal' distribution. The point here was that you needed to use a distribution that generates clusters of data and therefore you need a distribution that will have a peak density. A uniform distribution, for example, would not help since data would have the same density everywhere. The Gaussian distribution is a good example of unimodal distribution but you also need to know there are others. For example, the Gamma and Weibull distributions are interesting because they can be parametrised to generate various forms of distributions (that, unlike the Gaussian distribution, can be skewed). Remember that not everything is normally/Gaussian distributed!

Please see code `sem4_q1.{py,m}`. If the data is taken from a unimodal distribution then the best choice of k should be 1. However, remember that the algorithm is oblivious to the underlying distribution therefore if you ask it to find $k = 5$ clusters, it will find 5 clusters. This is likely to impact your generalisation.

Question 2

Please see code `sem4_q2.{py,m}`. In this question we simulate having two clusters. Here, you should play with the parameters (the mean mu and the standard deviation sigma) in order to create diverse scenarios. One would be where the two clusters are clearly separated. In that case, you should find that $k = 2$ is the best value and that the prediction by the k-means algorithm is very good. When using higher values than k, you will find that the k-means algorithm will once again artificially split the data. Remember, it has no idea what the underlying distribution is. If, on the other hand, you use only 1 cluster, then the most likely scenario is that it will come up with a centroid that will be in the middle of the two clusters. This is a very bad result for two reasons. First, it considers that all the points are from the same class. Second, it would classify any point that is close to the centroid as very likely to belong to the data. This would be very wrong and would lead to very poor generalisation.

Now, consider making the clusters closer or with larger standard deviations so that the clusters now overlap. Obviously k-means will continue returning the number of clusters you asked for but the quality of prediction will be very poor. The decision boundary (which is based on allocating points to the nearest centroids) means that you will have a linear decision boundary, irrespective of the actual class membership. You should not forget this is an unsupervised method.

Question 3

Please see code `sem4_q3.py,m`. This is just a generalisation of the above to 2D data, however, by manipulating the shape of the blobs you can get greater understanding of the assumptions behind the k-means algorithm. For example, when using Euclidean distance, the intrinsic shape of a cluster (e.g., whether it is elongated due to having correlations) will not play any role in determining class membership. Then, even with suitable separation between clusters, it is perfectly possible for members of one of the clusters to be predicted as belonging to another cluster. The following page provides some useful illustration.

Any question/comment, please email me or post on the forum!