Algorithms Design and Analysis

Clustering

Rayyan ul Haq,
Syed Ammar Ahmed,
Shehryar Mughal,
Rizwan Niaz,
Bilal Mohiuddin

Report presented for the course
CS 412 Fall 2019

Computer Science
Habib University
Karachi, Pakistan
6^{th} December, 2019

1 Problem

The problem this report is highlighting is that of clustering. This means that grouping observations together in a cluster that share similarities and hence creating k defined clusters. This is an optimization problem as you optimize the placement of an observation in a cluster which may vary throughout different algorithms.

2 Introduction

Clustering is an idea that is used in a lot of places. For example we have certain items that have features such as color, shape, etc. The idea of clustering them shows their similarities with the elements within the cluster and dissimilarity with elements from other clusters. To approach this problem we will consider five different algorithms and hence show their individual peculiarities and comparative analysis.

3 Approaches

3.1 K Means

The idea of K-means clustering revolves around its literal meaning of having k means in your observations and thus k clusters. This works in manner that it starts with k random points from the observations and consider them to be the centroids of k clusters, respectively. It then assigns the obervations to the clusters which they are closest to. Now the k centroids are updated and each centroid is now calculated from within its respective cluster. Each observation is again compared to every centroid and hence again assigned a cluster. This is continued for a constant number of times such as c. Lets say we have n observations. The time complexity of this algorithm is in $O(cnk)$ which is consequently $O(n)$. Such is so because we are comparing each observation with each centroid or mean hence nk comparison and this is done c times thus $O(nck)$.

3.2 Hierarchical (agglomerative)

This algorithm is based on the idea of taking a bottom up approach. The algorithm initially assumes that there are n clusters given that there are n observations. It then merges the two clusters which are the closest. This is done through maintaining a similarity matrix or distance matrix. Hence if we had initially n clusters we now have $n-1$ clusters. We repeat this process till we get our desired number of clusters. To get optimum number of clusters considering the distribution of observations, we use dendograms. The time complexity of this algorithm is in $O(n^3)$. Since we have to perform n iterations and in each iteration, we need to update the similarity matrix and restore the matrix as we will be comparing each cluster with every other cluster thus $O(n^3)$.

3.3 Divisive

The Divisive algorithm builds upon the ideas of the previous two algorithms. The Divisive approach starts out by assuming that there are all of the observations are in the same cluster. It then splits the cluster into two separate clusters. So if we have n observations then we start out by assuming all of them are in the same cluster. We split it into two more clusters each containing ¡ n observations. Now we take the mean squared distance between all the points of each of the clusters and once again split the cluster that has the higher mean squared distance between all of its points. Now we take the three clusters and split the one with the highest mean squared distance between all of its points. We do this until we have k number of clusters. In order to actually split the clusters into two we can use any number methods. For our case we used K means to split the cluster into two clusters at each step. The Divisive approach uses The K means algorithm at a total of k number of times, however, because we are splitting it in two, the time taken to do each split is $O(2nc)$ Since we are doing this k number of times, the time complexity of the algorithm on average is $O(2knc)$. The Divisive algorithms running time varies, depending on the distribution of the clusters. If for instance a cluster is split such that one cluster contains $n-1$ points while the other contains 1 point, The time taken to do this would be $O(2nc)$. The time taken to divide the larger of the two resultant clusters is thus $O(2(n-1)c)$ if we were to divide this in the worst possible way even further, the time complexity would be $O(2(n-2)c)$ if this is repeated for a large enough value of k then the worst case time complexity of the divisive approach is $O(2kn^2c)$. Consequently, if the clusters were to be split completely evenly at each step then the best case time complexity would be $O(2klog(n)c)$. The Divisive algorithm performs the best in the case where the clusters are spread far apart and have low variance within them and have circular distributions.

3.4 Gaussian Mixture Model

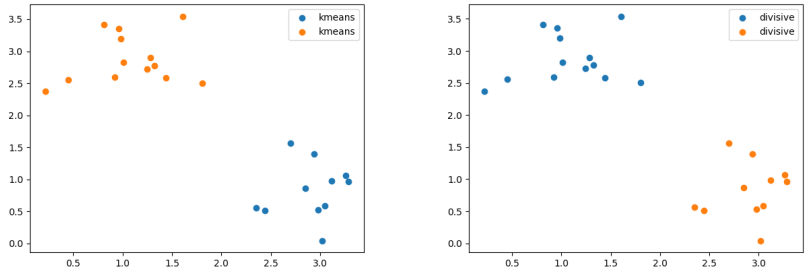
K-means is an algorithm that computes circular clusters. This is because K-means does not take into account the variance of our data. Because of this, while we do not always get appropriate results. Gaussian mixture models assume that, depending on the value of k that we set, we will have k Gaussian distributions that correspond to the number of clusters in our data set. Since we are now dealing with Gaussian distributions, we will still have the mean of our cluster but now we also have the liberty of introducing variance and co variance into our model. This gives us more freedom than k means as now instead of having exclusively circular clusters, we can now have elliptical clusters and because of this, we will have better approximations of probable clusters in the data set. We first assume k random Gaussian's that define the k clusters, then using Bayes theorem and expectation and maximisation algorithm, we find our new mean and new variance that define our clusters. We perform this a certain number of times, in our case 40, until we get the desired result.

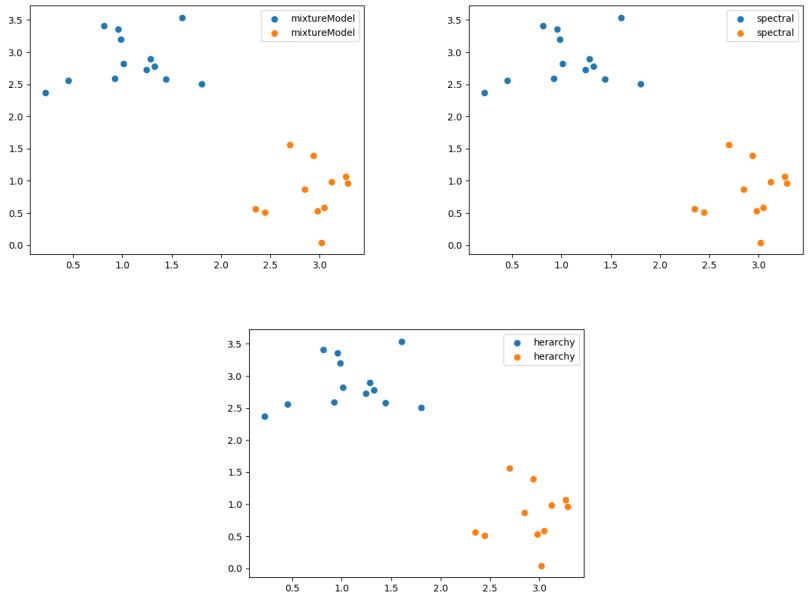
3.5 Spectral Clustering

Spectral Clustering is a clustering algorithm that is usually applied to a graph of connected nodes. To accurately represent our data of random points as a graph, we first set up a similarity matrix of size $n \times n$ where n is the number of points. Each position in the matrix is then filled with the euclidean distance between points. The similarity matrix is then used to create an adjacency matrix, where a value of 1 is added in a cell only if the corresponding euclidean distance is greater than a pre-defined filter. Once the adjacency matrix is created, we can proceed with drawing a graph for visualization purposes. To completely carry out spectral clustering though, we need to derive a degree matrix from our adjacency matrix. The degree matrix is obtained by summing each row of the adjacency matrix and then placing our result in the diagonal cell of the corresponding row. We then derive a Laplacian Matrix by subtracting the adjacency matrix from our degree matrix. Using the following property: *For a graph of x connected components, there exists x eigenvectors with an eigenvalue of 0*, we divide our points into x+1 clusters. In the case where the entire graph is connected but there are visible clusters that can be made, it is also possible to use the second largest eigenvalue and then divide the graph into clusters using k-means on the division created by the corresponding eigenvectors in order to get the next best clustering approximation.

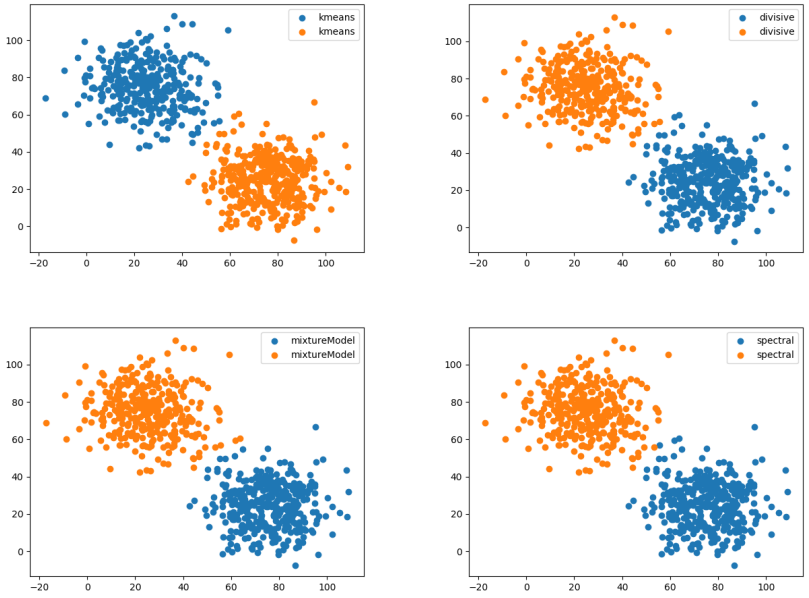
4 Experimentation

There is no objective measure for judging the quality of a cluster that is classifying some random data and therefore to provide some ground for judging our models we generated the data using normal distribution. This involved randomly generating the x and y coordinates about some mean and using the same standard deviation for both to generate circular clusters. The models after design were run on varying number of clusters and points to see which combinations resulted in somewhat the same clustering. The results of several of these are shown.
Small number of points with two clusters

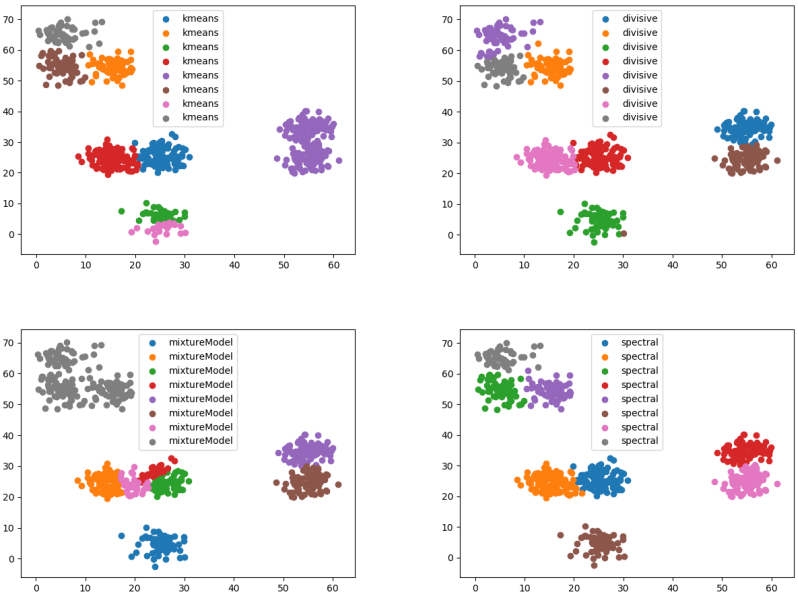




Large number of points with two clusters



Large number of points with eight clusters



Small number of points with twenty clusters

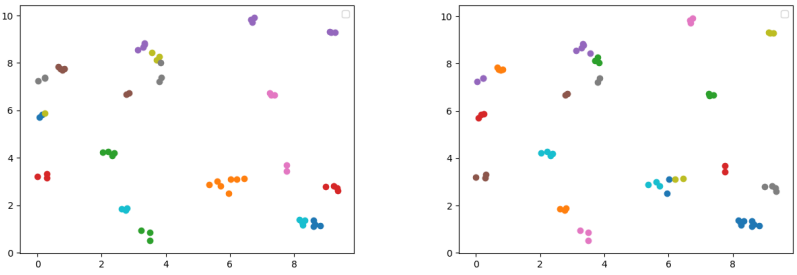


Figure 1: K-Means and Divisive Clustering

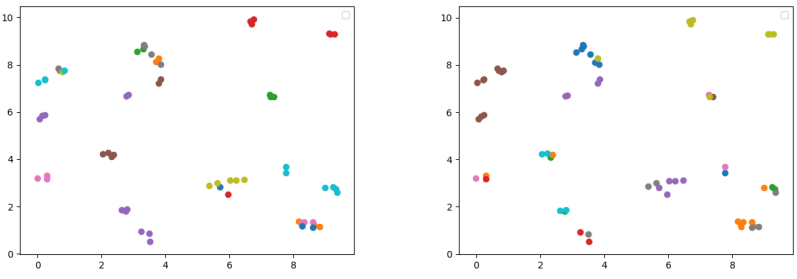


Figure 2: Gaussian Mixture Model and Spectral Clustering

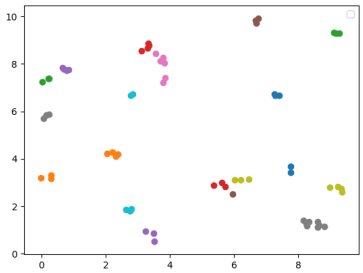


Figure 3: Hierarchical Clustering

Based on the above we concluded that two clusters is the case where all of the models perform exactly the same classification and this is the number of clusters we will use to judge the time complexity of our algorithms. Before that however let us try a few different cases where by literature some of the algorithms are expected to perform better clustering than the others.
In the following scenario, data was generated through two normal distributions having the standard deviation of x twice that of y. Thus producing somewhat elliptical instead of circular clusters. The performance of the models follow

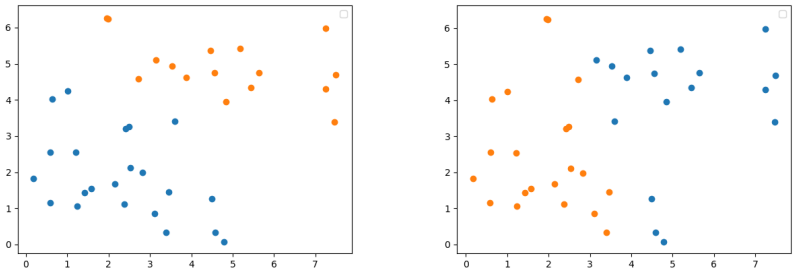


Figure 4: K-Means and Divisive Clustering

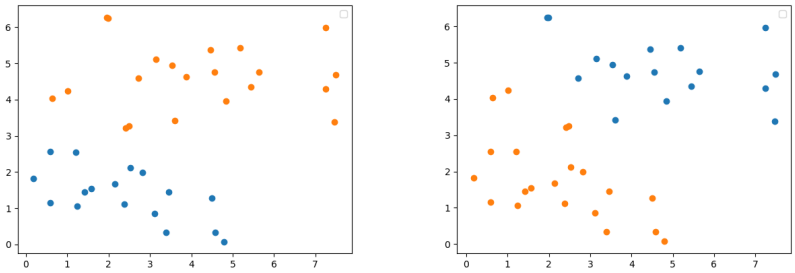


Figure 5: Gaussian Mixture Model and Spectral Clustering

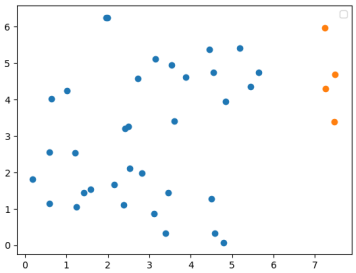


Figure 6: Hierarchical Clustering

From here we can see that the Gaussian Mixture Model which incorporates co variance matrices between x and y is able to correctly classify all the data points with spectral being a close second.
Another test case follows

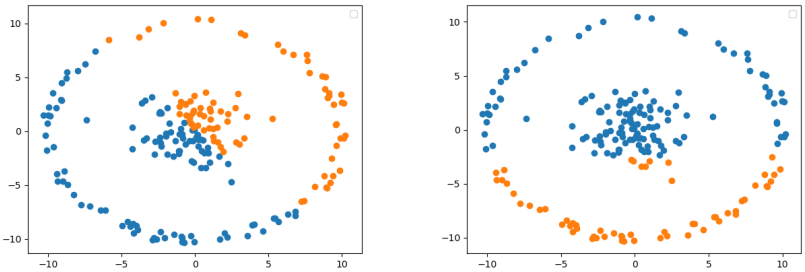


Figure 7: K-Means and Divisive Clustering

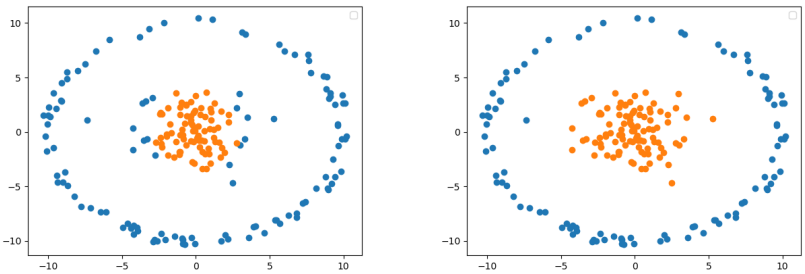


Figure 8: Gaussian Mixture Model and Spectral Clustering

This test case is shinning point of spectral clustering although Gaussian is able to perform fairly reasonable as well. The other algorithms which use euclidean distance as their base measure are unable to tackle the problems at all. We will now perform time complexity analysis using the number of clusters as two as previously established.

5 Time Complexity Analysis

The time complexity of the all of the algorithms follows. The hierarchical, as expected, has much worse time complexity compared to any of the others as can be seem from the first two figures and is thus removed from analysis for larger number of points in the next figures.

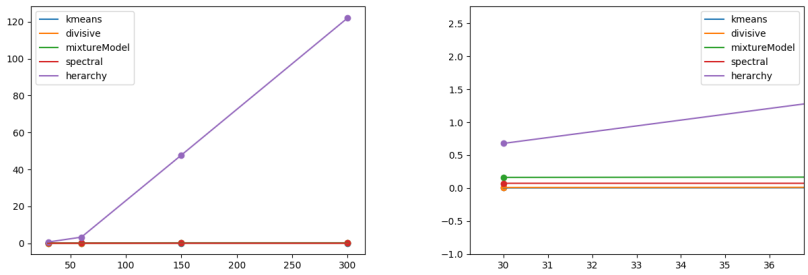


Figure 9: Time/s vs Number of points

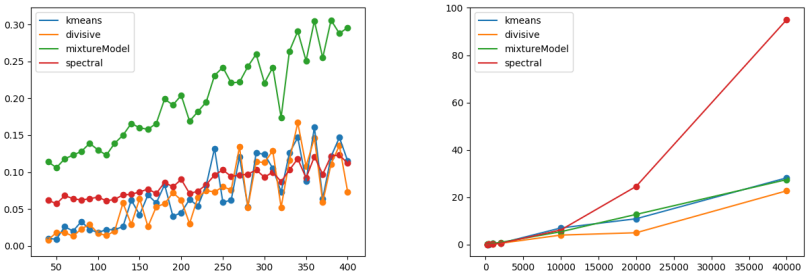


Figure 10: Time/s vs Number of points

For a small number of points that is up to 400 it appeared that the Gaussian Mixture Model has the worst complexity with other three taking the same time although varying widely because they incorporate randomness at some point in their algorithms. However when the number of points significantly increases we see that it is in fact spectral that has the worst complexity and where as Gaussian and K-means take approximately the same time and divisive consistently performs best.

6 Conclusion

In conclusion, the algorithm we use depends highly on our data set. For he most part, divisive clustering performs the best but given a specific type of data set, as in the case where we tested spectral clustering, it fails. Depending on what we want from our final result, we would use a different algorithm. But, even

with that fact, there are still some algorithms that perform better than others both in terms of complexity and usually the output such as divisive being better than K-means which is something to keep in mind.