Trace Tidwell

11/15/16

CpSc 605 – Assignment 5

Implementation of K-Means Algorithm in Python

In this assignment, we are implementing the K-Means algorithm in Python. The datasets used to test the implementation are the Two Ellipses dataset and the Iris dataset. The Two Ellipses dataset contains 2000 observations with 2 features each, and the Iris dataset contains 150 observations with 4 features each. This particular implementation of the K-Means algorithm takes in a dataset as a dataframe, the desired number of clusters, an error threshold, and a maximum number of iterations. The function returns the dataset with the cluster label appended and a summary dataframe with the final epsilon, number of iterations, and item count for each cluster label. The function also prints out a pairplot, which pairs all of the features with each other and plots them against one another, with the cluster labels indicated. First we’ll discuss the generic K-Means algorithm, then we’ll discuss its application on the two datasets. The general formula for the K-Means algorithm is shown in Formula 1 as:

(Formula 1)

For a set of n observations (x1, x2,…, xn), the data are divided into k sets (S1, S2,…, Sk). The goal is to minimize the distance from each point in the cluster to the cluster center, µi. As previously mentioned, this version of the algorithm accepts as inputs a dataframe, the number of clusters into which to split the dataset, a maximum error threshold, and a maximum number of iterations for the function to run. The last two inputs are stopping criteria. The first step is to initialize the centroids. The dataset is copied and randomly shuffled. From there it is broken into k groups, where k is the input to the function. Then the centroids for each of the k groups is found by taking the mean value of each feature. The next step is to calculate the distances from each data point to each of the k centroids. The distance here is the Euclidean distance, defined in Formula 2 as:

(Formula 2)

where p and q are two points in Euclidean n-space. A matrix of indices is then returned, with the index indicating to which group or cluster each point belongs. From the newly grouped dataset, the centroids are once again calculated. The error ε (epsilon) is calculated as the sum of the Euclidean distances between the new centroids and the old centroids, and then the old centroids are replaced with the new ones. The process is then iterated through until one of the stopping criterion is met. In this case, it will run until the maximum number of iterations has occurred or until ε is less than the specified amount, whichever comes first. Upon reaching one of the stopping criteria, the dataset is updated with the cluster labels for each data point. A summary table is made which includes the number of iterations and ε, along with the count of observations in each cluster group. Finally, a pairplot is made to help visualize the effectiveness of the clustering algorithm. The updated dataset, the summary, and the plot are all returned.

Since the Two Ellipses dataset contains only 2 features and 2 clusters, we’ll start with it. Using the given parameters of k = 2, ε < 0.001 and number of iterations < 20, we can call the KMeans function. Table 1 below shows the summary table generated by the function.

|  |  |  |  |
| --- | --- | --- | --- |
| ε (epsilon) | Iterations | Group 0 | Group 1 |
| 0.0 | 2 | 1000 | 1000 |

Table 1: Summary Data from Two Ellipses K-Means Clustering

We can see that the algorithm very quickly and cleanly separated the data into two evenly sized groups. By viewing the Pairplot graph in Figure 1, we can see that the data were already separated into two distinct groups, so we’d expect the algorithm to work as it did.

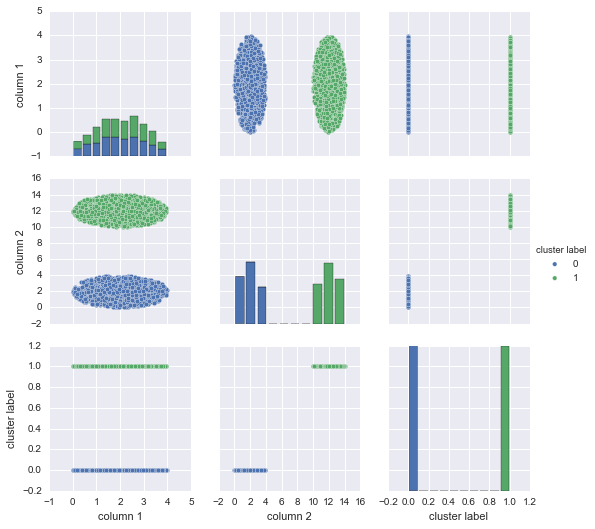


Figure1: Two Ellipses Pairplot

The second application of the K-Means algorithm was a bit more complicated, as we had 4 features and 3 clusters. Once again the stopping criteria were given as ε < 0.001 and number of iterations < 20, but this time k=3. The summary data for the run are listed below in Table 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ε (epsilon) | Iterations | Group 0 | Group 1 | Group 2 |
| 1.339226 | 19 | 50 | 62 | 38 |

Table 2: Summary Data from Iris K-Means Clustering

For this dataset the ε is still rather high, and we can see that we reached our maximum number of iterations. I decided to experiment with the maximum number of iterations to see if the algorithm would converge on a set of centroids that would reduce ε. However, even after setting the maximum to 190, ε and the group counts remained the same, meaning that for this dataset, this is as close as it’s going to get. Looking at the Pairwise graph for the dataset, it becomes a bit more clear as to why this is the case.



Figure 2: Iris Pairplot

There is much less distinction between the groups than with the Two Ellipses datasets. Group 0 is fairly easy to separate from the whole dataset, but Groups 1 and 2 tend to overlap quite a bit. This is also demonstrated by our cluster counts. Each group should have 50 entries, but Group 1 has 62 while Group 2 has only 38. Because the correct classes were also given for this dataset, we can calculate the accuracy of the algorithm. Since 16 observations were incorrectly identified, that means 134 out of 150 were correctly identified, or 89.33%.

Considering the K-Means algorithm is the simplest clustering algorithm, the overall performance exhibited is very good. We correctly grouped all of the observations from the Two Ellipses dataset (based on inspection), and we correctly grouped ~90% of the Iris dataset.