HW4

David Schmidle

Miner Name: Luckytiger

Part-1 Latest Score: 0.65

Part-2 Latest Score: 0.60

K-Means Clustering

A diagram of red dots

Description automatically generated

Preprocessing:

Preprocessing for the Iris and Starlight Curves dataset followed the same general formula: Normalization / Standardization followed by dimensionality reduction via PCA.

A graph with green dots

Description automatically generatedFigure 1 and Figure 2 analyze the first two features of the Iris dataset: Sepal Width & Sepal Length. For a K-Means algorithm that computes the mean of all data points in each cluster to evaluate the next centroid, it is necessary for the data to be preprocessed correctly. In this case, we scale unit variance and remove the mean for all four features in our dataset. From the figure visualizations, which don’t include all four features of the Iris dataset, we can visually draw two clusters much more easily after standardization.

Figure 1. Iris Dataset Plotted by Sepal Width & Length before Preprocessing

PCA was chosen as a feature-reduction technique because it is unsupervised (in both Iris and Starlight we are not given the labels for the data).

Figure 2. Iris Dataset Plotted by Sepal Width & Length after Preprocessing

\

K-Means Algorithm Implementation:

The K-Means Algorithm is implemented as follows: K (the number of clusters) is provided. Upon calling *fit\_predict,* which supplies the clustering model with the data it needs to calculate the clusters, the centroids are initialized with randomly selected data points. Unfortunately, this sequentially causes some issues in clustering since the K-Means algorithm is potentially sensitive to intial conditions depending on the data. Consider figures 3 & 4 that use different starting centroids. Utilizing an initalization algorithm such as K-Means++ would be beneficial in clustering data that is sensitive to initial conditions.

A diagram of a number of dots

Description automatically generatedA diagram of different colored dots

Description automatically generated

Figure 4. Initial Iteration of Centroids using random selection (seed=1)

Figure 3. Initial Iteration of Centroids using random selection (seed=14)

The stopping condition of our K-Means algorithm is measured by pairwise comparison of the calculated centroids after each iteration. NumPy has a method, *allclose,* which is usedfor this implementation which compares each value with a certain amount of error in mind. This error can be changed upon initialization of KMeans. An alternative condition for stopping in a K-Means algorithm is pairwise comparison of all the points in each cluster; if between iterations these sets of points fail to change, the clustering algorithm will halt. This implementation was avoided as this comparison at each iteration can be extremely computationally intensive with an increasing number of data points. However, it is likely that this solution would yield more accurate clustering in terms of classification.

At each iteration of the K-Means algorithm, each point in the dataset is assigned to a cluster by calculating Euclidean distance to the centroid. This is usually a very computationally intensive process, especially for larger datasets; however, using NumPy’s broadcasting techniques we can significantly reduce the computation time required, even for larger datasets such as Starlight Curves. This requires adding a new axis (dimension) for calculation, so NumPy has compatible dimension shapes.

A graph with a line

Description automatically generated

Naturally, after each datapoint has been assigned a cluster, each centroid is updated by calculating the mean of each cluster. There is an edge case where no data-points are assigned to a specific cluster, resulting in no average available to calculate; in this case, we simply choose another random point in the dataset and continue iteration.

Lastly, we can calculate the sum of squares error, which is the distance between each data point and its associated centroid in each cluster.

Figure 5. Sum of Squares Error as K increases

As seen in Figure 5, the SSE decreases as the number of K-Clusters increases. This intuitively makes sense as more clusters spread across our dataset means more points will have closer centroids. This is a strong indication that our K-Means implementation is working as intended.

Results:

A diagram of different colored dots

Description automatically generatedSince K-Means clustering is unsupervised and our dataset is unlabeled. It is difficult to test the accuracy of our K-Means algorithm as a classification model since we lack the label of each instance. In figure 6 and 7 we can visualize the clustering algorithm at its final iteration (denoted at the top of the scatter plot).

Of course, to visualize the K-Means clustering algorithms, there must be some dimensionality reduction techniques that reduce the features down to 2. As noted earlier, the decomposition technique used was PCA, which projects the data to a lower dimensional space. It is projected on the 2 dimensions that explain the most variance in the data; A diagram of a cluster of dots

Description automatically generatednaturally this process exhibits some error.

Figure 6. Final iteration of K-Means Clustering on the Iris dataset with PCA ncomponents = 2, seed=3

Additionally, among all iterations using different seed values (which can be changed upon initialization of the KMeans algorithm), rarely did the KMeans algorithm surpass 15 iterations before reaching the stopping condition. However, if K was increased, such as in calculating the Objective function vs K, the number of iterations increased.

Figure 7. Final iteration of K-Means Clustering on the Starlight Curves dataset with PCA ncomponents=2, seed=1

A graph with a blue line

Description automatically generatedIn figure 8 we can see the sum of squares error for the starlight dataset, where we measure SSE as a function of K. As with the Iris dataset, we can see here that the SSE continues to decrease as expected with higher values of K.

Figure 8. Sum of Squares Error for Starlight Dataset