CSCE 474/874: Introduction to Data Mining
Spring 2021

Assignment No. 3 March 02, 2021

Documentation of Contributions

Please use this form to document the contributions of each team member to the group effort. You may either sign electronically or sign and scan it. In either case, you must submit this document along with all other elements of the assignment.

Furthermore, you confirm that the work submitted is entirely your own and that you understand the consequences of plagiarism as specified in the Course Syllabus and UNL CSE Academic Integrity Policy.

Name	Contribution %	Contributions	Signature & Date
Justin Ho	50	Wrote code for the implementation of the algorithm and worked on the report. Generate data for comparison with Weka	Justin Ho (03/14/2021)
Kolby Johnson	50	Wrote code for the implementation of the algorithm and worked on the report. Comparative analysis on Weka	Kolby Johnson (03/14/2021)

1 Program Design

1.1 General Overview

The program is written in `c++` and the entry point of the program is in `main.cpp`.

The implementation of the clustering algorithm is within the class named `kmeans_clustering` and the source code for this class and its methods are in the file `kmeans.cpp`. The parser for the input file can be found in `parser.cpp` and the output writer is in `writer.cpp`. The data can be found in the `data` directory.

1.2 Implementation of K-Means

The K-Means algorithm accepts the number of clusters k, the threshold for change in sum of the distance ϵ and the maximum number of iterations. The last two of the fields are the terminating conditions. The initial position of the centroids are initialized randomly to the points in the dataset.

The K-Means algorithm abstracts the implementation of this algorithm by providing a public method `execute()`, which runs the following line of code:

```
1       for (int i = 0; i < this->max_iterations; i++)
2       {
3           double change_SSE = kmeans_clustering::compute_SSE();
4           if(change_SSE < this->epsilon){
5               break;
6           }
7           kmeans_clustering::assign_cluster();
8           kmeans_clustering::update_centroids();
9       }
```

The program terminates when it hits one of the two conditions. In line 3 for the computation of SSE calculates the euclidean distances for each of the points with respect to their assigned cluster and returns the total change of the distance. Line 7 reassigns the points to the clusters with the nearest centroids and line 8 updates the centroids for each cluster.

2 Function Plots

2.1 Number of Clusters vs Time

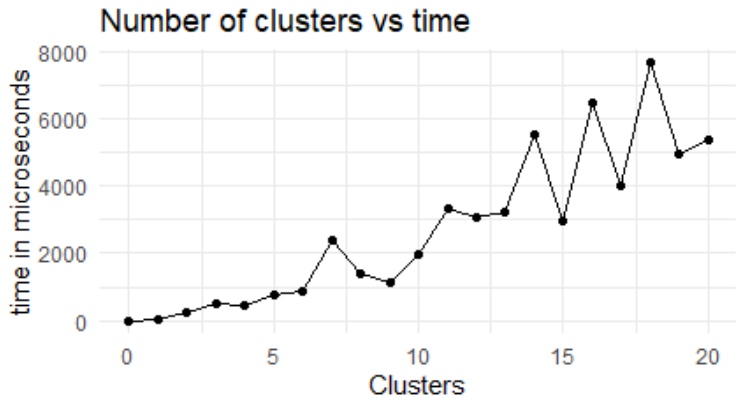

Figure 1: plot of the number of clusters vs time

The following graph is obtained by using different values of k and measuring the difference in running time in microseconds with the file `blob.csv`.

2.2 Number of Dimensions vs Time

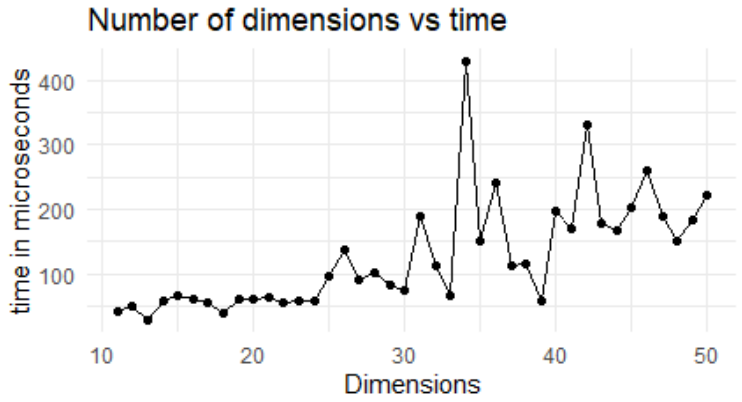

Figure 2: plot of the number of dimensions vs time

The following graph is obtained by using different dimensions of the data `blob.csv` and measuring the difference in running time in microseconds. The dimensions

are obtained by multiplying the original x value of the data with a random number generated from a uniform distribution $\in [0, 1]$.

2.3 Number of Transactions vs Time

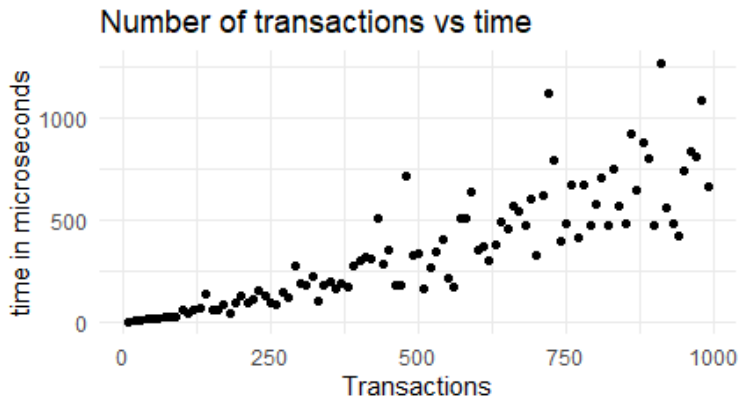

Figure 3: plot of the number of transactions vs time

The following graphs is obtained by subsetting the dataset `blobs.csv` by increments of 10 and measure the time taken in microseconds.

As you can see from the plots as the time increases so do the other variables. Since the algorithm's performance is characterized by $O(n \times k \times I \times d)$, which is observed in the graphs above.

3 Goodness Plot

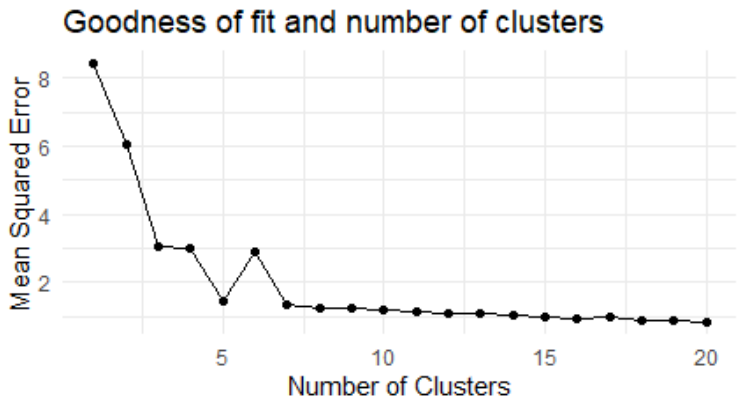

Figure 4: plot of the number of clusters vs the goodness of fit

As you can see by the plot when the number of clusters increases the MSE goes down. This shows that as the number of clusters increases the mean distance goes down with each cluster causing the MSE to decrease.

4 Weka

4.1 Iris Graphs

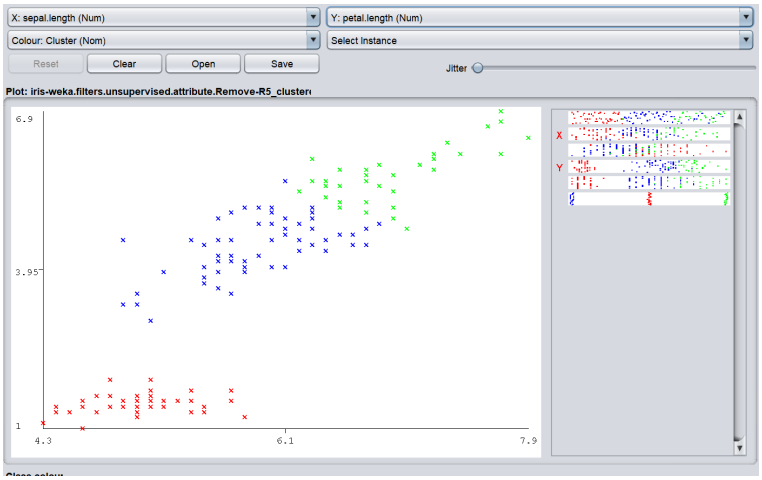

Figure 5: Weka Clusters for Iris data set

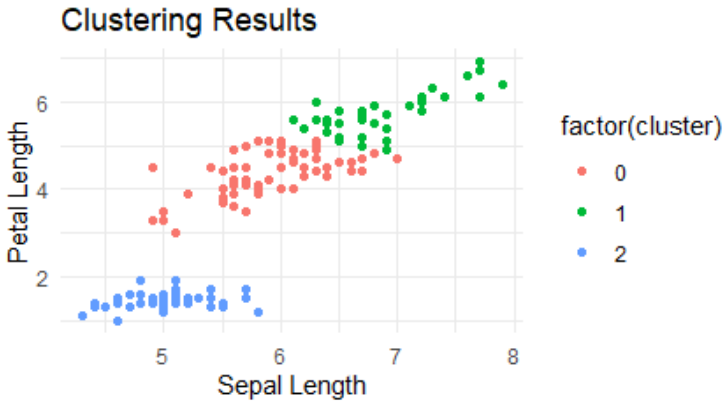

Figure 6: Clusters generated by our algorithm for Iris data set

The graphs above are the results of clustering with Weka and our program. The dataset is obtained from the famous Iris dataset.

4.2 Moon Graphs

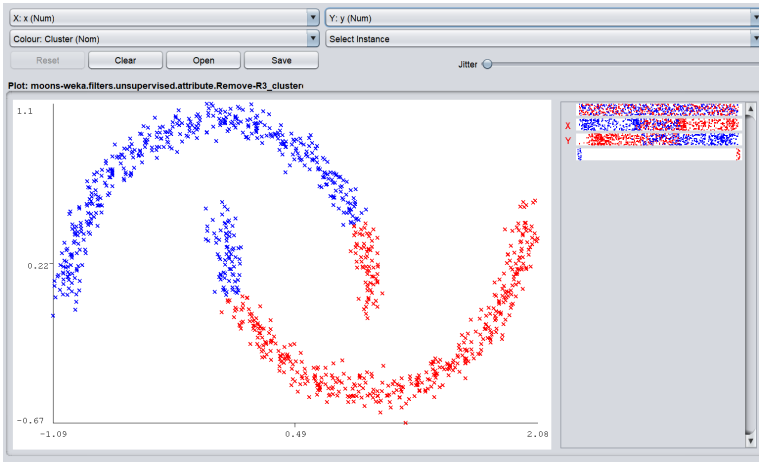

Figure 7: Weka clusters for Moons data set

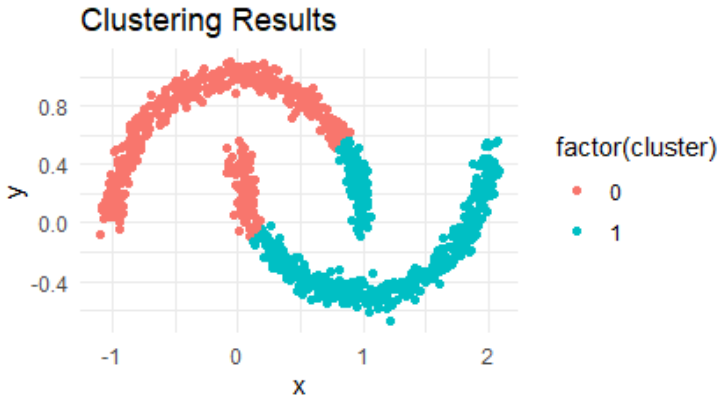

Figure 8: Clusters generated by our algorithm for Moons data set

The graphs above are the results of clustering with Weka and our program. The dataset is obtained from generating Moon graphs in `python`.

4.3 Blob Graphs

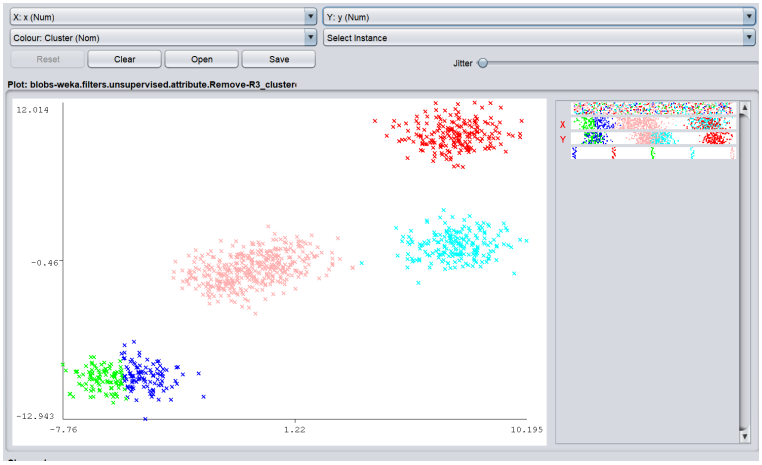

Figure 9: Weka clusters for Blobs data set

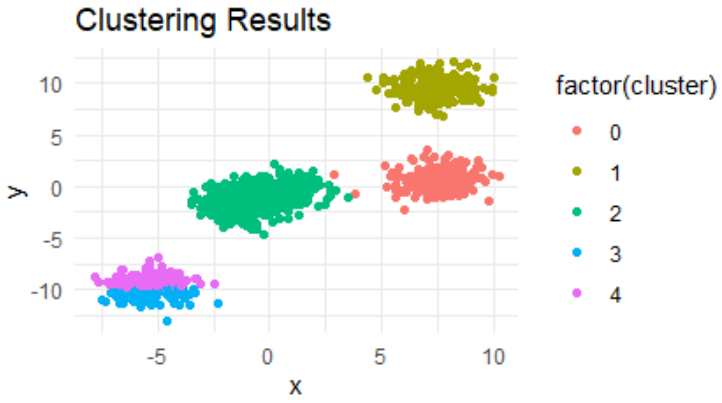

Figure 10: Clusters generated by our algorithm for Blob data set

The graphs above are the results of clustering with Weka and our program. The dataset is obtained from generating Blob graphs in `python`.

4.4 Graphs Comparison

As you can see the clusters generated by our algorithm are very similar. The only one were there is a major difference is in the blobs clusters where Weka decided to split the bottom group side to side, our algorithm split it up and down. In terms of SSE our algorithm was very close to Weka's only being off by a slight amount.