Determining Best Clustering Technique Which Mimics Real World Visuals on WWLLN Lightning Data

Alejandro Herrera, Ben van Oostendorp, Aeon Williams

Dr. Barnabas Bede, Dr. Yilin Wu, Dr. Jeremy Thomas, Dr. Natalia Solorzano

DigiPen Institute of Technology

**Introduction**

There are many different techniques for clustering data, but the question on which method best illustrates what we see in the real world with lightning strikes originating from the same cloud/group of clouds was not fully explored. We chose 2 commonly used clustering techniques and one slightly less common for clustering lightning to inspect the differences and compare with what we would expect them to look like in the real world.

Keywords: clustering, lightning, kmeans, dbscan, fcm

**Data and Methodology**

The data used was obtained from the World Wide Lightning Location Network (WWLLN), which is lightning strokes from all over the world. The type of lightning WWLLN detects is the electromagnetic radiation from the return stroke. The data was then processed by the Naval Research Lab with storm centered coordinates. Each point of data now has a longitude, latitude, date and time, and distance from the center of the storm. From here, we did a lot of visualization of the data, by loading it up, selecting the data form every 30 minutes, and then began to cluster with K-Means, DBSCAN, and FCM. Each clustering algorithm was setup with data in two different ways: partition the data based on distance to the storm and bucketize them, then cluster within the buckets, and taking the whole data, 30 minutes worth, and running the algorithm.

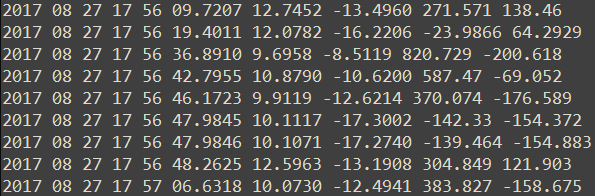


Figure : WWLLN Data with storm centered coordinates

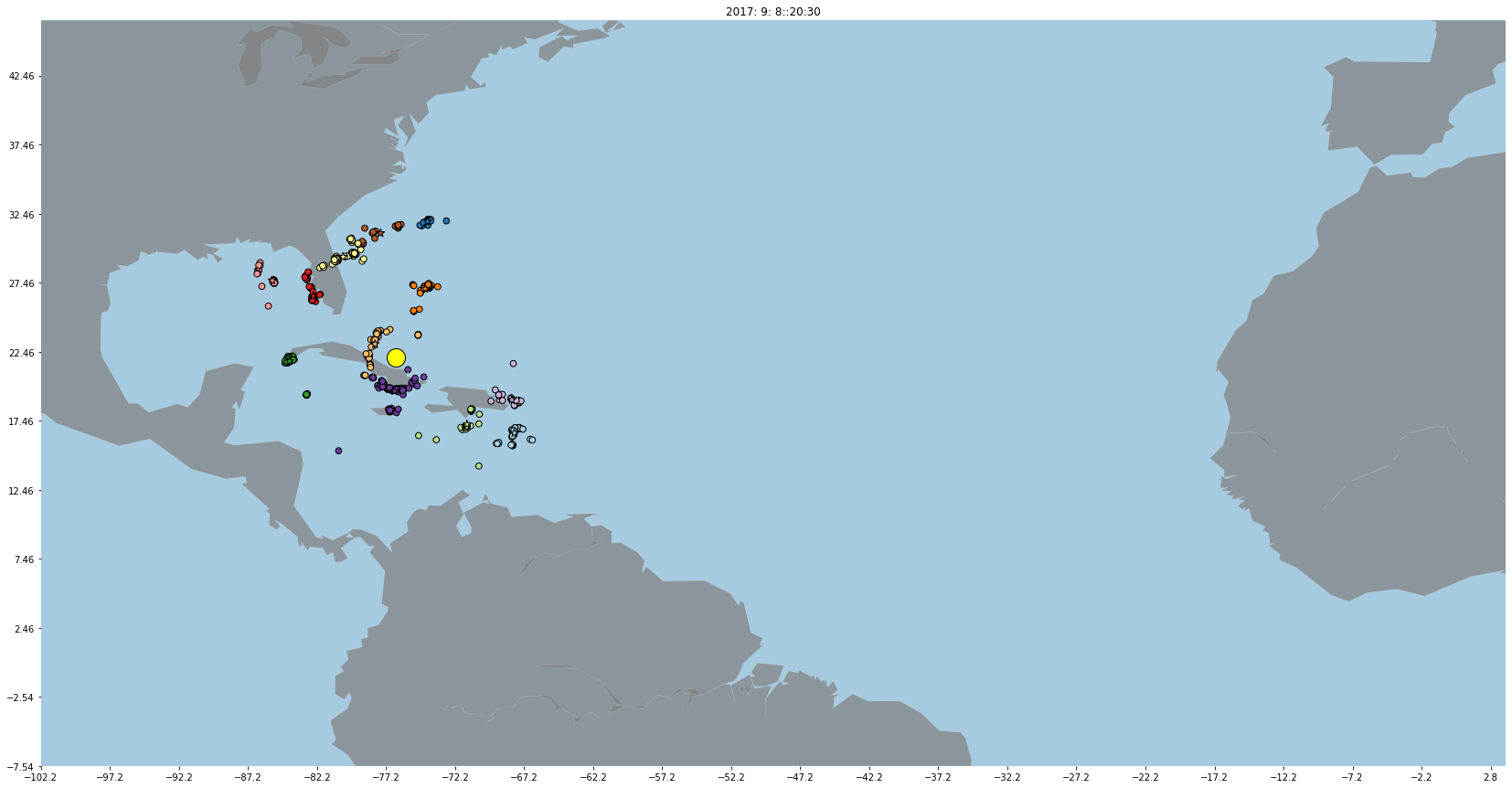


Figure 2: WWLLN Data Plotted

**K-Means**

K-Means is a method of clustering that aims to partition n observations in k clusters, where each observation belongs to a cluster, and the center, or centroid, of the cluster is the mean of all points in the cluster. K-Means sets to minimize the variance of each cluster by optimizing the squared errors. It creates clusters based on *k* number of clusters.

**DBSCAN**

DBSCAN, or density based spatial clustering of applications with noise, is a density-based clustering algorithm that groups n observations based on density and closeness of observations in space. It creates clusters based on input parameters such as the maximum distance a point can be from a neighborhood and how many observations it requires within the neighborhood to be labelled a core observation. Observations that do not fit into the criteria are labelled as noise or outliers.

**Fuzzy C-Means**

Fuzzy C-Means (FCM) is very similar to K-Means in that it too aims to partition n observations in k clusters, but the membership of each observation to each cluster is not 1 (entirely belonging to) or 0 (does not belong). In FCM, observations can exist in multiple clusters to varying degrees, where a higher membership value means it exists more in the center of the cluster than observations with lower membership. One strong point to using FCM is that we can figure out if clusters are well defined with the membership matrix, since we can calculate the entropy per row. This gives us information on how “in” each cluster each point is. The lower this number, the better defined the cluster. The algorithm we used was developed by J.C. Dunn and improved by J.V Bezdek.

**Comparison of the Clustering Algorithms**

Once we ran each clustering algorithm on the data, it was noticeable that there were differences between each algorithm. We then looked at each of the algorithms pair wise to notice differences, compare results on the same data, and ultimately look to answer the question of which algorithm would best emulate the results we see in real clouds.

**K-Means vs DBSCAN**

Start the comparison of K-Means and DBSCAN here.

**K-Means vs Fuzzy C-Means**

These two clustering algorithms are very similar and can have almost identical cluster composition depending on the parameters used – this is immediately visually apparent when plotting the results. Because we are clustering geographical data instead of trying to predict labels, we focused on the cluster composition similarity instead of the label similarity. The Mouse dataset was used for some initial testing, before moving on to the lightning dataset.

We observed that although we could tell the clusters were almost identical sometimes, the difference in order of cluster creation led to false statistical data that they were dissimilar, as seen below.

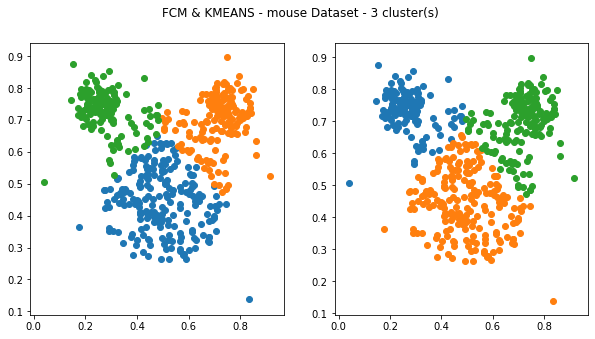


Figure : FCM & Kmeans 0% similarity

To fix this issue, the labels were manually adjusted based on corresponding cluster center locations to examine the cluster composition similarity ratings. This changed the 0% similarity plot above to be the 98% similarity plot below.

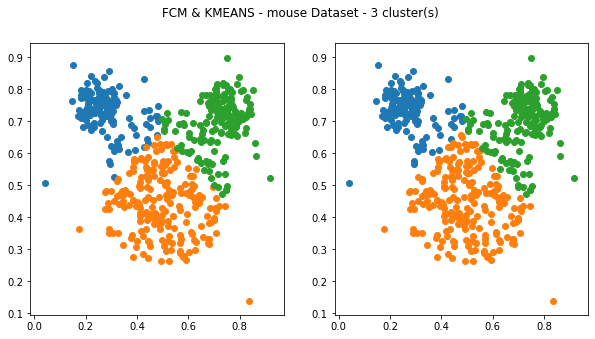


Figure : FCM & Kmeans Adjusted Label

Overall, the cluster results were similar with very few clusters, and exponentially become dissimilar with increased cluster count, as seen below.

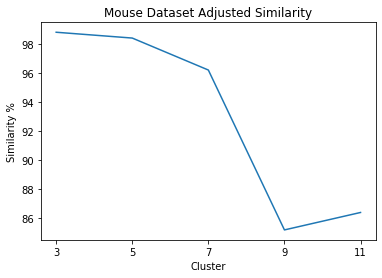
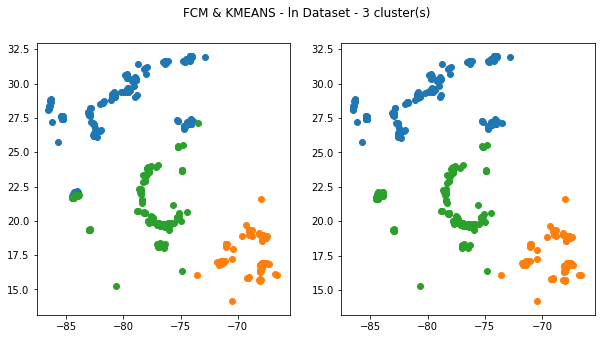
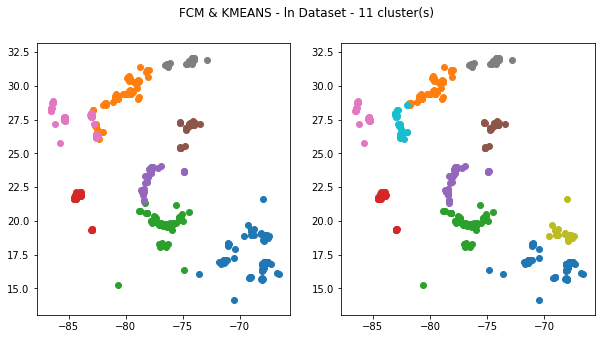


Figure : Mouse dataset similarity plot

The lightning dataset produced a similar pattern. There was a 97% similarity in pattern with three clusters, and an 86% similarity with eleven clusters.

Figure : Lighting dataset cluster plot



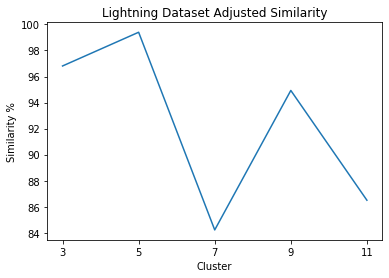


Figure : Lighting dataset cluster similarity plot

FCM has a significantly slower computation time compared to KMeans, taking minutes instead of milliseconds to perform operations. We expected this, but it makes FCM much less appealing in a generic context because Kmeans is very similar but extremely faster. FCM is still good for edge cases, such as observing the entropy of data with many dimensions. It also clustered the lightning data a little bit more cleanly with eleven clusters compared to Kmeans, so there can be further testing done with even more clusters to see how they compare in that context.

**Fuzzy C-Means vs DBSCAN**

The comparison of FCM and DBSCAN was very similar to the results that K-Means and DBSCAN produced. However, FCM still holds the data for calculating the entropy, which can be useful, but given that we are mostly looking for the structure of the clusters rather than how well defined they are, this may not be the most useful. The figures below are a comparison of the “combination” of FCM and DBSCAN. First DBSCAN was run on the data and plotted, and then the number of clusters determined by DBSCAN was then passed to FCM and plotted. From the images, we can still see that DBSCAN performs better, but perhaps some further work to use calculated cluster centers from DBSCAN and give them to FCM as the initial centers and then running the algorithm might yield better results.

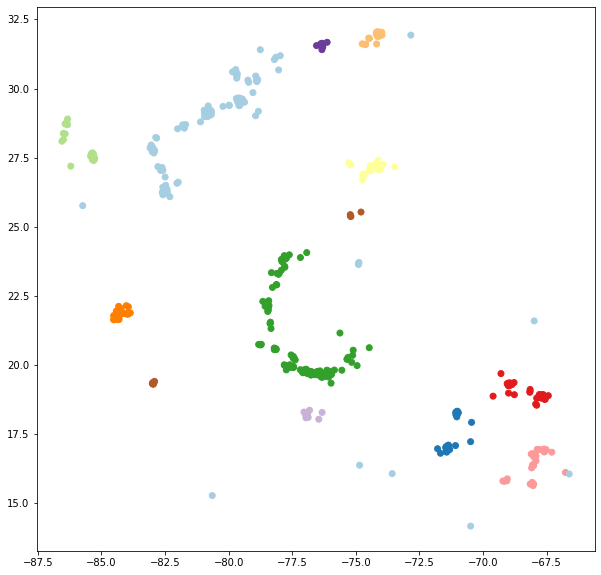


Figure 8: DBSCAN on WWLLN data; Hurricane Irma

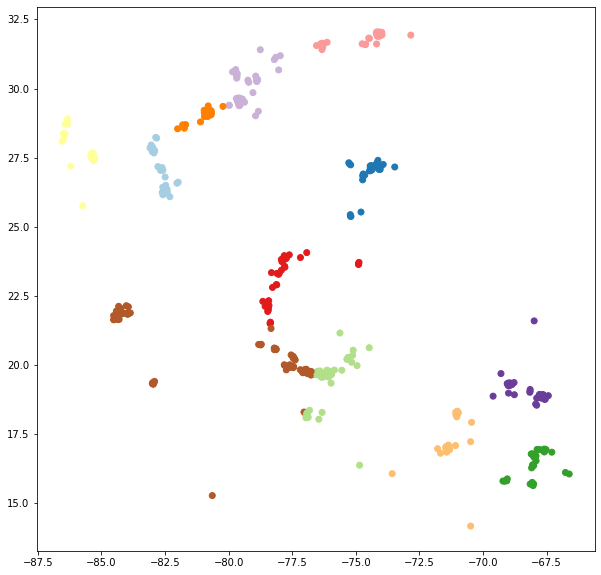


Figure 9: FCM on WWLLN data; Hurricane Irma

****

Figure 10: FCM Membership Matrix Entropy (lower means better defined clusters)

**Results**

After running and comparing each algorithm, we see that DBSCAN has the better real-world results that we expect from cluster structure. When it came to comparing FCM and K-Means, the results showed that the two algorithms performed very similarly, which was to be expected. While it is possible to generate the similar cluster structure/composition from K-Means and FCM, it results in overfitting the algorithms to a specific data splice and would have to be changed for each splice.

**Conclusions**

We see that DBSCAN has the best results for future structure, but this is just for the WWLLN data set. When we begin to incorporate more data, namely the GOES-16 GLM data and the microwave imaging from NASA, we will most likely dip into using FCM and DBSCAN in tandem, especially for the microwave data when forming composite images.