Final Project 3: Geo-Location Clustering Using the k-Means Algorithm

BENJAMIN CHOI, JESSIE KOROVIN, and NICK MURRAY

An approach to big data geo-location clustering using Lloyd's algorithm (the naive k-means algorithm).

Additional Key Words and Phrases: big data, geo-location, clustering, k-means

ACM Reference Format:
Benjamin Choi, Jessie Korovin, and Nick Murray. 2019. Final Project 3: Geo-Location Clustering Using the k-Means Algorithm. 1, 1 (October 2019), 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Our project is about clustering using the k-means algorithm. Clustering is a data analysis approach used to learn about a dataset's structure. The main idea is that we should implement some algorithm that identifies groups in the data where, hopefully, each point in a subgroup is relatively similar to another point in its subgroup, but is also relatively different from another point outside of its subgroup. For example, if we clustered WiFi-enabled electronic devices based on screen height and width, we would probably see three individual clusters for smartphones, tablets, and computers. The algorithm we'll use in clustering is called k-means. K-means is a distance (user-defined) based method that updates the location of k (user-defined) centroids until they converge. The goal of the k-means algorithm is to assign points in the dataset to a cluster such that the distance between each point and a cluster's centroid (corresponding to the arithmetic mean of data points in the cluster) is the minimum.

Clustering is important for data-analytics in the real world. It's a powerful tool that's able to solve a wide variety of problems. For example, when a company (with a chain of stores) is expanding rapidly, it might need to build warehouses to store items, but might only be able to afford 5 warehouses. One way the firm could determine where to put these warehouses would be to use k-means clustering to determine optimal locations for the warehouses. The company would set k=5 (and could use a distance measure like euclidean distance), and cluster on the location of all their stores. The 5 final centroid locations would be optimal places to put warehouses because then the sum of the squared distance between each point (store) in a cluster and the cluster's centroid (warehouse) would be the minimum. The scope of clustering is beyond just determining locations, it can also be used in identification algorithms (identifying malicious URLs), marketing (finding customers with similar purchasing behaviors), and even in the analysis of social networks.

As to why clustering is important in the context of big data and cloud computing, the amount of interesting information in this world is growing at a rapid pace. According to ecology.com, there are "four births each second of every day." If we were interested in finding similarities based on the heights and weights of every living person, even if we didn't account for people being born

Authors' address: Benjamin Choi; Jessie Korovin; Nick Murray.

XXXX-XXXX/2019/10-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

right now, we'd still need to examine 7.8 billion data points. Clustering may be able to give us insights on similarities between people of different heights and weights, but we'd never be able to learn anything on these 7.8 billion data points without cloud computing. In the first place, learning algorithms (such as k-means) are iterative, so the runtime without using a cluster of computers would be terrible. Also, for more complex datasets, we won't be able to fit the information on a single computer. In summary, k-means clustering is important in big data because it provides us a way to try and learn about datasets that would be too big to analyze otherwise.

2 OUR APPROACH

2.1 Obtaining the Data

We first needed to download all three datasets that we'll be using. This meant downloading the files for the device status data, synthetic location data, and DBpedia location data.

For our final dataset, we decided to examine a DEA dataset which shows the geo-location of every opiate pill transaction recorded from 2006-2012. We discovered this data from this[1] Washington Post article that examined the DEA's pain pill database. With this dataset, we wanted to explore the best locations for rehabilitation centers in New Jersey. Although it would be interesting to examine the country as a whole, we decided to focus on New Jersey because even that one state's data amounted to ~2 GB made up of ~4.27 million data points. Additionally, all 3 of our group members grew up in New Jersey, so we felt like the impact of our findings would hit closest to home.

2.2 Data Preprocessing

While the synthetic location data and DBpedia data did not need to be processed, we needed to process both the device status data and our chosen DEA dataset. The goal of our data preprocessing efforts was to format all of our datasets such that each would have latitude and longitude data as its first two features, split by either whitespace or a comma.

To do this for the raw synthetic location and DEA datasets, we used Spark to take in the original data, split each line by the relevant delimiter for the data point, filtered out any records that did not parse correctly or had a latitude and longitude values of 0, and included all of the other relevant fields we might need. For the synthetic location data, we included fields like the date, model, and device ID. However, for the DEA data, we only needed the latitude and longitude data.

2.3 Implementing Helper Functions

In order to implement our k-means algorithm, we needed seven helper functions:

(1) *closestPoint*: a function that takes in a point, a list of centroids, and a distance parameter (euclidean or great circle) and outputs the centroid closest to the point using the given distance measure
(2) *averagePoints*: a function that takes in a list of points, averages them, and outputs the average point
(3) *euclideanDistance*: a function that takes in two points and outputs the euclidean distance between them
(4) *greatCircleDistance*: a function that takes in two points and outputs the great circle distance between them
(5) *averageDifferences*: a function that takes in two arrays of points and a distance measure and outputs the average element-wise distance between the points of the two arrays, using the given distance measure

[1]https://www.washingtonpost.com/graphics/2019/investigations/dea-pain-pill-database/?fbclid=IwAR3eE7cHwrzLjMo5L5V9jkBTei16-TwymFEjZ9q4ebdgqaAVKzzs8D4t5-E

(6) *geoToCartesian*: a function that takes in a point of latitude and longitude pairs and outputs a point converted to (x, y, z) Cartesian coordinates
(7) *cartesianToGeo*: a function that takes in a point of (x, y, z) pairs and outputs a point converted to latitude and longitude pairs

2.4 Implementing K-Means in Spark

Our k-means algorithm takes in four parameters: k, the distance measure (either euclidean or great circle), the location of the input file, and where to output the results.

The algorithm first takes in the input file, converts each data point to a pair of latitude and longitude coordinates, and then persists the data (as we will refer to it constantly). From the latitude and longitude coordinates RDD, the initial k centroids are set by randomly sampling k points from the RDD without replacement. From there, for each coordinate, we see which centroid is closest to the point using *closestPoint*, calculate the new centroids by averaging all of the points assigned to the centroid using *averagePoints*, and do so until convergence (which we check for by feeding in the old centroids and new centroids to *averageDifferences* and seeing if the result is less than our convergence distance parameter (which we decided to set to 0.1).

After our k-means algorithm runs, we know the latitude and longitude data of each centroid and which data points belong to which centroid, which we output to the output location.

2.5 Visualizing Our Results

We decided to plot our results using Python's matplotlib library. We used a scatter plot and the library handled plotting each point by its coordinates, which we first convert from latitude and longitude pairs to (x, y) Cartesian coordinates. We set the opacity to be very low (0.03) so that the points can overlap, thus allowing us to intuit density depending on the darkness of the color. We used a colormap to differentiate the clusters, with each cluster receiving a different color tag from darkest to lightest.

The different points overlay each other, and we added an image of the map area in the background with the max and min of the coordinates to match up with the plotted points. Lastly we plotted the centroids, making them bigger and a different color (blue) to emphasize the difference.

2.6 The Choices We Made

Although we allow the user to specify the k and distance measure, we decided to choose 0.1 as the convergence distance, decided to persist the latitude and longitude RDD specifically, and decided to use Python's matplotlib library to visualize our data. Here, we just wanted to explore why we made these decisions.

In regards to the convergence criteria, we chose 0.1 as it seemed like the best balance between runtime (making sure the algorithm does converge at some point) and performance (making sure the result of the algorithm appears reasonable). Experimenting with other values for the convergence criteria led to either a huge sacrifice in runtime for minimal gains in performance or a huge sacrifice in performance for minimal gains in runtime, thus we ultimately settled on 0.1. In regards to our choice to persist the latitude and longitude RDD, we made this choice as this is the only RDD that we would need to access repeatedly without modification, thus making it the ideal candidate for persistence. Finally, we decided to use Python's matplotlib library as it is an extremely robust tool and one which our group had prior experience with.

3 PSEUDO CLUSTER IMPLEMENTATION

3.1 Results

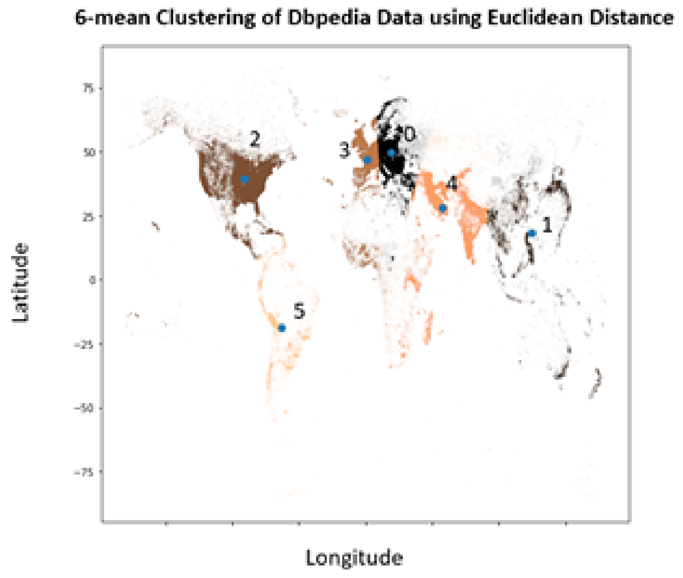


Fig. 1. DBpedia Worldwide Clustered Results

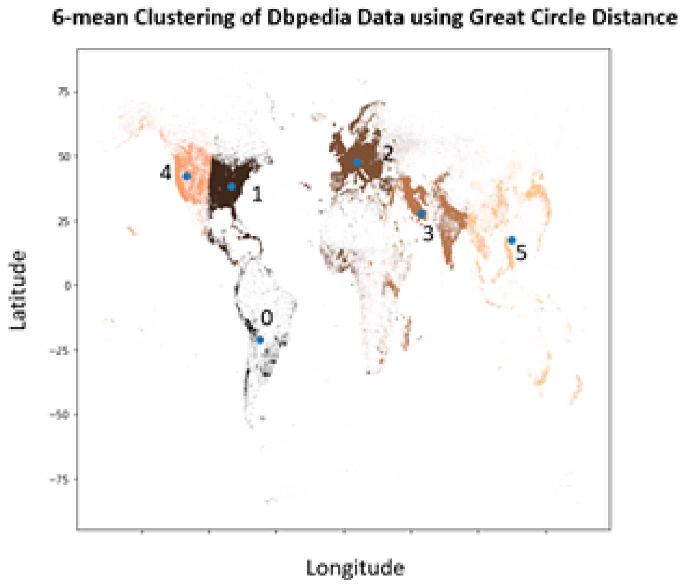

6-mean Clustering of Dbpedia Data using Great Circle Distance
Latitude
Longitude
75
50
25
0
-25
-50
-75
0
1
2
3
4
5

Fig. 2. DBpedia Worldwide Clustered Results

We had a successful implementation of our Spark k-means clustering that ran fairly quickly, and we visualized the final results on the 2-D figures above using matplotlib.

Using the entire DBPedia Location Data as our pseudo-cluster data source, we calculated clusters for k=6 for data which had geolocations spanning the majority of the world. k=6 makes sense as a choice because of the volume of the data, and how the clusters mostly centered around the 6 different continents represented. Each cluster could represent a large zone of coverage for a majority of users.

We visualized the cluster centers using euclidean distance in Figure 1 and great circle distance in Figure 2. The largest difference is that the clusters are actually different. For euclidean distance, clusters 0 and 3 are split over Europe while great circle distance instead splits two cluster centers (1 and 4) over North America. In this case, using euclidean distance led to much quicker convergence, but great circle distance performed more accurately and made more sense to us from a visual inspection. Overall, we felt that great circle distance was the preferred distance measure in this case because of how far this dataset spans. As the dataset covers points from across the globe, it becomes increasingly important to account for the curvature of the Earth while clustering (something great circle distance accounts for, but euclidean distance does not). However, if the dataset spanned a much smaller area (like one state in the US, for example), using euclidean distance might be better as the curvature of the Earth might not be as big of an issue and using euclidean distance does not require the extra time-intensive trig operations needed to calculate the great circle distance.

3.2 Runtime Analysis

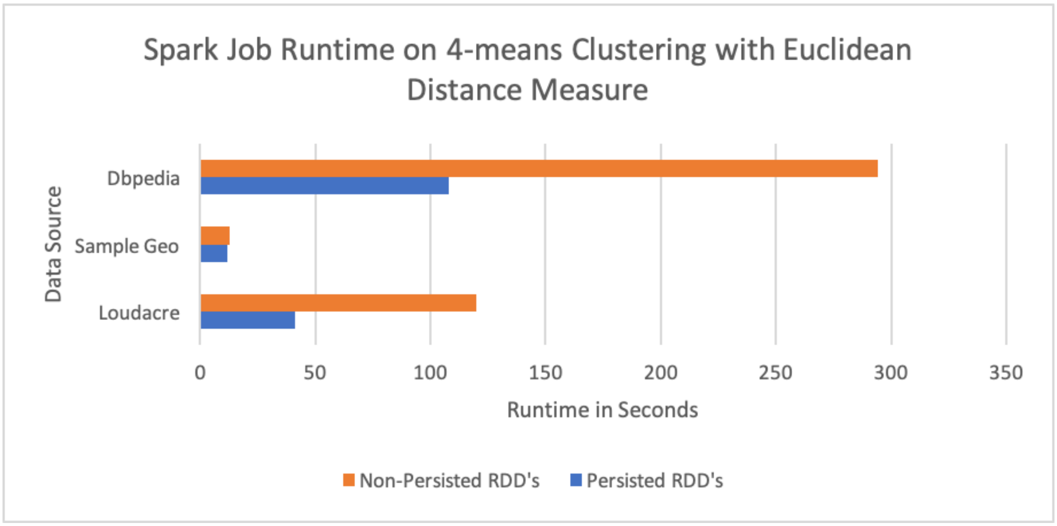


Fig. 3. Spark Job Runtime Analysis for our three data sources.

Looking at the results from our 6 Spark job runs, we can see a couple patterns based off of which data source we used, and if we persisted the data for each iteration.

DBpedia was the largest data source, and consistently took longer on its job runs than the other two smaller data sources. The second pattern that is key to note in our runtime analysis is that persisting the intermediate data significantly cut down on runtime for the larger data source as they scaled. It did not make a big difference for the smallest data source (Sample Geo), but for the other two it more than halved the total runtime. In the non-persisted RDD Spark jobs, the location data needs to be recalculated for each subsequent iteration, which unnecessarily extends the total runtime for each job.

3.3 What's Next - Big Data Application

After implementing our psuedo-cluster successfully, our next step was to find a big dataset to run on Amazon EMR, which we would then need to preprocess into the proper format of latitude-longitude pairs. We can then visualize this data in the same way as before on a 2-D plot.

The next largest part of the project will also be setting up and executing our Spark Job on the cloud. We must configure our Amazon account, and submit our k means file with our uploaded Big Data set to get our final results.

4 BIG DATA APPLICATION: DETERMINING OPTIMAL LOCATIONS FOR DRUG ABUSE TREATMENT CENTERS

4.1 Dataset and Application

The dataset we used was created by the Drug Enforcement Administration (DEA), and can be found at the Washington Post's *website* (filtered by state).[2] The total size of this dataset was ~4.27 million transactions, and the **runtime** took ~2 minutes (120 seconds) on Amazon EMR.

The dataset tracks the path of every opioid sold in the U.S. from 2006 to 2012, accounting for nearly 380 million pain pill transactions. We're particularly interested in clustering the opioid buyers' addresses (latitude and longitude). **We decided that we would focus our efforts on determining the optimal locations (represented by centroids) for drug misuse treatment centers.** As all three members of our group are NJ residents, we decided to do the clustering for NJ opioid order data.

The original dataset contained about 20 features per row, which included information about the type of drug, the reporter (seller) who got the transaction, the address of the buyer, the quantity, and the manufacturer of the drug. For preprocessing, we dropped everything except the buyer's zipcode, which is what we used to determine our lat-lon pairs for our Spark implementation.

We created a dictionary object that held all of the zip codes for New Jersey, and mapped it to a lat-lon pair. We avoided using exact addresses because converting all of these into an exact lat-long would've taken considerably more time (and required the use of an external API). Using zip codes represented a faster and cheaper solution, with only a slight decrease in accuracy. As the zip codes represented a fairly small area, we were able to get a good approximation of the lat-lon pairs.

Misuse of prescription opioids is a prevalent issue. While most people use prescription opioids as intended, according to drugabuse.gov, "in 2017 an estimated 18 million people (more than 6 percent of those aged 12 and older) have misused [opiods] at least once." Currently, there are programs set in place to combat opioid misuse, such as prescription drug monitoring programs, which seek to identify patients who are likely to be misusing based on frequency of visits and other factors.

As seen in the figure below, drug abuse in New Jersey is on the rise.

While drug monitoring programs are helpful, it's still insufficient to solve a problem where, according to catalyst.phrma.org, "54.2 percent of prescription drug misuse cases result from the abuser procuring the drug free from a friend or relative." Right now, it is hard to dream of completely stopping the drug misuse problem, but we believe that adding centers dedicated to helping people overcome their addictions would reduce the rate of overdose death.

Using k-means clustering, with euclidean distance, we can optimally place treatment centers so that help for abusers is easily accessible. In addition, we also wanted to determine which clusters are most populated to see which areas are most prone to drug misuse, thus providing useful information to the government. Preventing drug misuse is costly, and if we could give the government information about where the most drug misuse is occurring, they could determine which areas need more funding.

[2]https://www.washingtonpost.com/graphics/2019/investigations/dea-pain-pill-database/

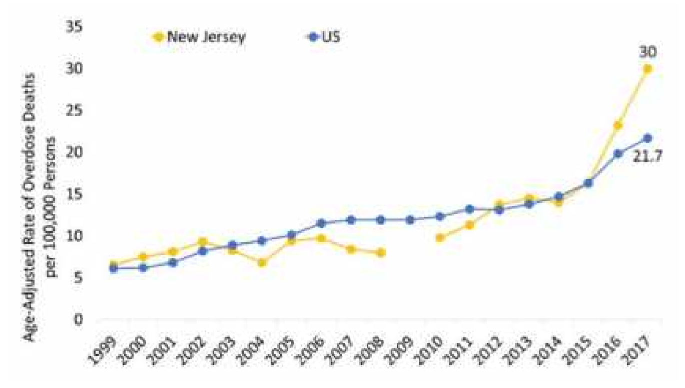


Fig. 4. Drug overdose deaths, rate per 100,000 persons, in the US; and New Jersey. Source: CDC WONDER. *Circumstances in New Jersey for the year 2009 have resulted in unusually low death counts, therefore this data point has been omitted.

4.2 Results/Discussion

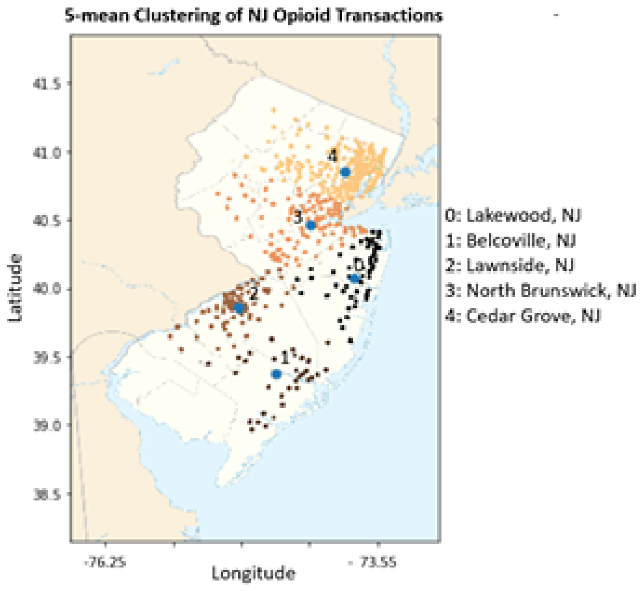
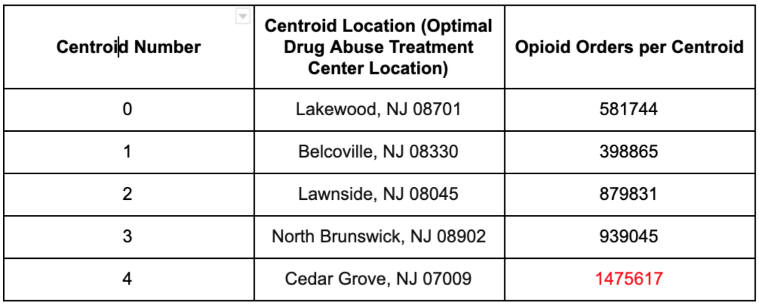


Fig. 5. There are 5 clusters here ranging from North to South Jersey, with the largest cluster in North Jersey where a large percentage of the population lives.

After running the algorithm several times, we determined that setting k=5 with euclidean distance provided a sufficient clustering solution. In terms of the distance measure, we felt that euclidean distance was the best choice for this dataset as it only encompassed locations from one state, meaning that the curvature of the Earth was less of a concern (making the choice of great circle distance less useful). In terms of our choice of k, we see that it does well in splitting NJ into its most populated areas: north eastern, north central, east central, west central, and south NJ. We also



Centroid Number	Centroid Location (Optimal Drug Abuse Treatment Center Location)	Opioid Orders per Centroid
0	Lakewood, NJ 08701	581744
1	Belcoville, NJ 08330	398865
2	Lawnside, NJ 08045	879831
3	North Brunswick, NJ 08902	939045
4	Cedar Grove, NJ 07009	1475617

Fig. 6. Display of which centroids have the most opioid orders

think 5 drug treatment centers is reasonable given the scope of the opioid epidemic plaguing NJ. The 5 optimal locations are shown in the table above.

It's important to note that because of our use of zipcodes that many points are stacked on top of each other, so that the final visualization does not look as large as the 4 million transactions that are actually plotted on the chart. To accurately get a sense of the density of each cluster, we provided a table that shows how many transactions are actually contained within each cluster. In this case, cluster 0 has the least with almost 600,000, whereas cluster 4 in Northern NJ has almost 1.5 million drug transactions.

When looking at opioid orders per centroid, it makes sense that the most points were assigned to the centroid at Cedar Grove, NJ. According to the NJ news publication *The Patch*, 2 of the top 3 towns for prescription opioid abuse deaths are in Essex County NJ (which is where Cedar Grove is located). It's useful being able to see this data in detail, as we see that there are 3.6 times more opioid orders in the most densely populated cluster than in the least populated cluster. If there is a budget constraint for the government in fighting this crisis, by looking at the data they'd see that the Essex County area is most in need of help.

5 CONCLUSION

5.1 Final Conclusion

Overall, we're satisfied with the results of our k-means implementations and visualizations for the three small datasets and our large dataset. We're able to gain a lot of insight by using cloud computing to visualize datasets that are too big to analyze on a single machine. We're also able to acknowledge that managing runtime is very important, as if we had not been careful in our implementations (for example, not persisting RDDs that are used multiple times), we'd have to spend much longer getting results for our datasets.

5.2 Lessons Learned

We learned a couple of important lessons throughout completing this project. The first was about the reasons we would want to use different distance measures. Let's take great circle distance and euclidean distance as examples. Great circle distance performs better than euclidean distance when working with latitude and longitude points, but there is a runtime tradeoff because of the expensive trig operations needed to calculate the great circle distance. Throughout this project we saw that great circle distance is better for datasets that span huge distances, in order to account for the curvature of the Earth. If we have a dataset for only one small area (for example, our dataset with points just in NJ) it makes more sense to use euclidean distance because we can save a lot on runtime without losing too much performance. Another thing about k-means that we learned is that there's no best value of k for all datasets. It depends on the dataset and it is best to select a k value using trial and error (how well is the data clustered visually?). We also learned a lot of practical applications, such as visualizing data using matplotlib, preprocessing big data using spark/RDDs, how to use RDD persistence intelligently to get faster runtimes, and how to run Spark applications on real clutsers using Amazon EMR.

5.3 Future Work

After working through this project, our group thought of many things we'd like to do. For example, we could experiment with other distance measures to see if they help our clustering or make the algorithm faster. We would also like to learn how to visualize k-means using more than 2 features. As of now we only figured out how to plot in two dimensions, but it's apparent that (geo-location clustering aside) most clustering applications work with more features than just two. That said, we think learning how to work in more dimensions would allow us to solve more difficult and practical clustering problems. Additionally, in the future, we would also like to learn how to use k-means more generally (for example, we'd like to learn how to handle clustering strings rather than numbers).

ACKNOWLEDGMENTS

To the TAs, for all their help throughout the semester. And to Prof. Neumann, for an informative course and a good semester. :-)