Final Project Report: K-Means for Geo-Location Clustering in Spark

Ziqian Luo, Adrien Xie, James Chen, Jason Zhu
Washington University in St.Louis

December 14, 2018

Motivation

Clustering has many useful applications and the k-means algorithm is one of the approaches for clustering. As one of the unsupervised learning algorithms, k-means is fairly robust and general. Yet, it is very useful in data visualization and data interpretation. The team members were motivated to understand how k-means algorithm works and its application.

As for Apache Spark, it is a very powerful tool that overcomes many limitations of Hadoop MapReduce, and we are interested in understanding what the combination of Apache Spark and the k-means algorithm can do in large-scale geo-location data. The team members were particularly interested in the cloud execution of the k-means algorithm as Spark jobs. It is intriguing to understand how Spark framework supports the execution and how different hyperparameters would affect the results.

In order to study the k-means algorithm as well as Apache Spark, we first implemented the k-means algorithm and then submitted Spark jobs to Amazon EMR cluster, where we extracted different sizes of big datasets and fitted them into the k-means algorithms. Finally, we visualized results from clustering and analyzed our experimental outcomes with our comprehensions.

Project Preparation

Milestone 1

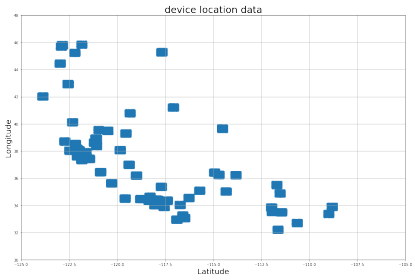
Milestone one consists mostly of project logistics, such as project topic and group member determination. In addition to project logistics, team member roles and project responsibility are assigned according to each individuals' preference.

Milestone 2

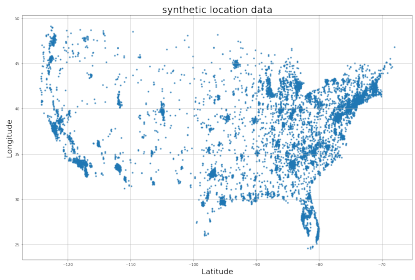
In milestone two, three datasets which would be used in our experiments were preprocessed. They are device status data, synthetic location data and DBpedia

location data.

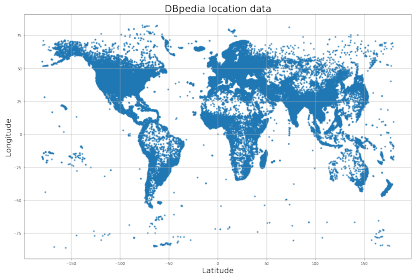
The device status data was preprocessed by applying data scrubbing for future processing. First, incorrectly parsed data entries were removed from the dataset and then the remaining was reordered as well as filtered out by zero-valued longitude or latitude. The extracted device status data contained 283480 records and each record contained five attributes. Afterward, the preprocessed data was stored as comma delimited text files on HDFS. For the synthetic location data and DBpedia location data, the longitude and latitude were given, therefore, no further preprocess action was required. We then plotted all three datasets so that we were able to interpret what the datasets mean through their geographic graphs. The visualization graphs are shown in Figure 1. We could notice that the area ranges of three datasets were different. Device location data focused on the west coast of U.S. while the synthetic location data expanded across the U.S. The DBpedia location data contains geolocation data points from the whole world.



(a) Device location



(b) Synthetic location



(c) DBpedia location

Figure 1: device location data

For final routing, Ziqian Luo did the programming both local and cloud as well as collected, analysis data and composed the final report. Adrien Xie deployed the AWS and did partial experiments on the cloud. He also helped to write up the final report. For James Chen, he was responsible for managing the SVN as well as work-flow.

Clustering Approach

In this project, we used k-means clustering to cluster our datasets. K-means clustering is a non-uniform vector quantization that aims to partition n observations into k clusters that each contains sets of points from n observations which have the nearest mean associate to each k centroids points. By providing heuristic algorithms or iteration limits, k-means clustering will converge to the local optimum.

K-means Steps and the Choice of *k*

Though there are many approaches to doing k-means clustering, we chose the iterative refinement approach for its simplicity and easy understanding. This approach has overall three stages. First, we need to sample k initial points from provided observations. Second, calculating the means from sampled k points with respect to all observations and evaluating the bias between the sampled k points and the calculated means. At last, refining sample points with the evaluation result and continuously computing the means until evaluation meets the convergence setting.

For mean calculation, it can be calculated by two different distance functions, Euclidean distance, and Great circle distance. Euclidean distance is a commonly used distance calculation algorithm that compute the straight line distance between two points within the Euclidean space. Euclidean distance is shown as the following function,

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{1}$$

where n is the number of observations we have in the dataset.

On the other hand, Great circle distance computes the shortest distance between two points on a sphere with function,

$$\begin{cases} \Delta\lambda = p_2 - q_2 \\ \Delta(p,q) = \arctan \frac{\sqrt{(\cos q_1 * \sin \Delta\lambda)^2 + (\cos p_1 * \sin q_1 + p_1 * \cos q_1 * \cos \Delta\lambda)^2}}{\sin p_1 * \sin q_1 + \cos p_1 * \cos q_1 * \cos \Delta\lambda} \end{cases} \tag{2}$$

where p_1, q_1 stands for latitude of each points and p_2, q_2 stands for longitude of each points.

For the evaluation, we use within-cluster sum of squares error as our heuristic function.

$$\sum_{x_i \in k} (x_i - \bar{x}) \tag{3}$$

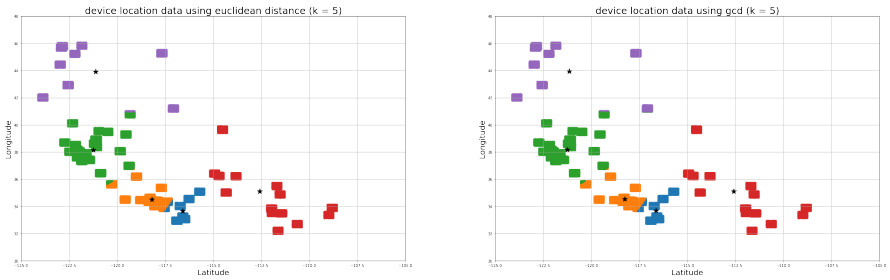
which k stands for our sampling data points and $\bar{x}$ is the mean of the points within each cluster associates with points in k.

Throughout the experiment, we experimented different numbers of k to find out the suitable k for each dataset as well as the difference performance and outcome between two different distance function under fixed k condition for different size of data.

Small Dataset on Pseudo Cluster

In order to understand the k-mean clustering, we first performed our experiment on a pseudo cluster using three different sizes of datasets which were introduced in previous sections. Through this experiment, we aimed to observe how different k values will affect the appearance of the data clusters also the difference between two distance functions. Since we wanted to observe the difference between two distance functions under a non-uniform algorithm, we fixed the seed for sampling initial data points in order to give them the same starting stage.

For the device dataset, we tried with $k = 5$ for k-means clustering and produced the following results respect to two distance functions.

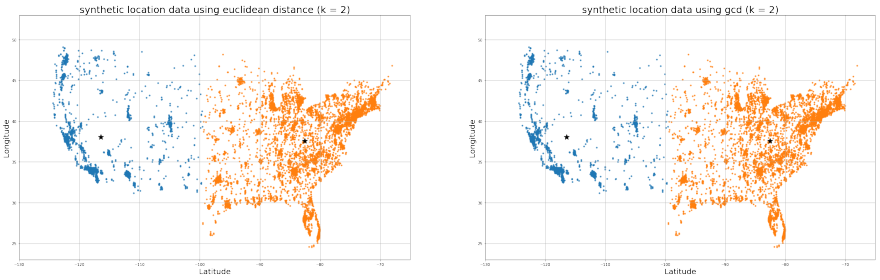


(a) euclidean distance (b) great circle distance

Figure 2: device location data

From Figure 2, we were able to see that two distance methods did not produce a noticeably visually difference when clustering this dataset. The geo-location points in this dataset located on the west coast of U.S. thus had a small range difference on both latitude and longitude. Therefore, distances calculated by two distance function will not have a huge difference. Also, the points are distributed in groups already, the sampled points would not have a great shift during the refinement process. Thus, two results were similar to each other.

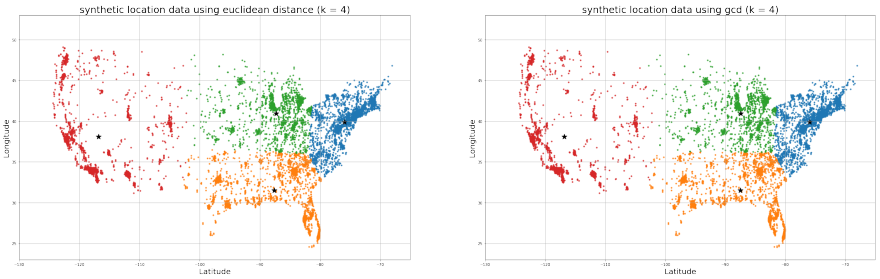
Afterward, we performed k-means clustering on synthetic location data with $k = 2$ and $k = 4$. This dataset composed of sparse geo-points all over the U.S. With $k = 2$, the outcome shown in Figure 3,



(a) euclidean distance (b) great circle distance

Figure 3: Synthetic location data (k = 2)

we observed that because of the uneven distribution of points between two coasts, densely on the east coast and sparsely on the west coast, two distance method again did not produce visually indistinguishable results. And with $k = 4$, shown in Figure 4,

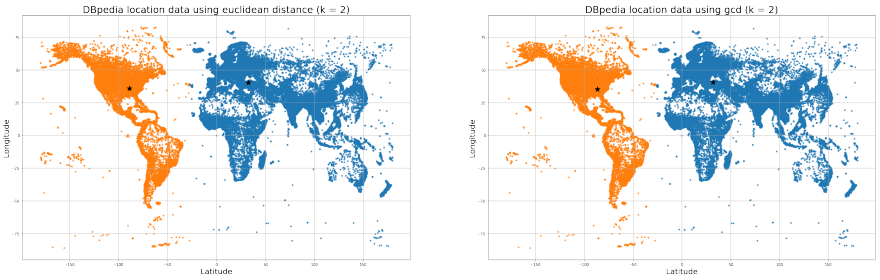


(a) euclidean distance (b) great circle distance

Figure 4: Synthetic location data (k = 4)

we discovered two new centroids on the east coast. The reason new centroids appeared on the east coast instead of west coast was due to the more density of points on the east coast. Therefore, points on the east coast could be characterized easier than the west coast. Thus, we could conclude that centroids would more likely to appear within a dense cluster of points than sparse one.

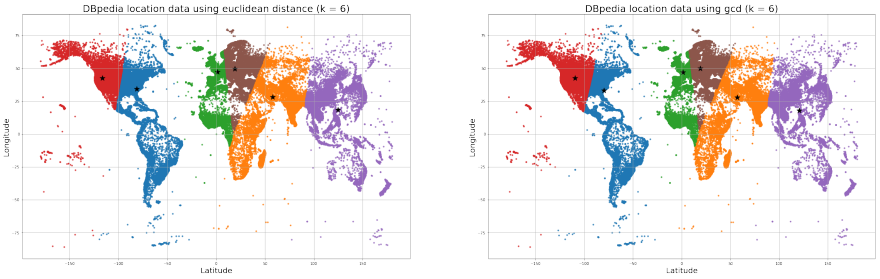
For the last dataset we experimented on our pseudo cluster, we chose three different values of k which are 2, 6 and 10. The reason we picked these three k values was that DBpedia location data was a fairly large dataset compared to the previous two which had over 400 thousand data points that crossed the whole world. We hoped to observed what different kinds of clusters would be produced by these k values.



(a) euclidean distance (b) great circle distance

Figure 5: DBpedia location data (k = 2)

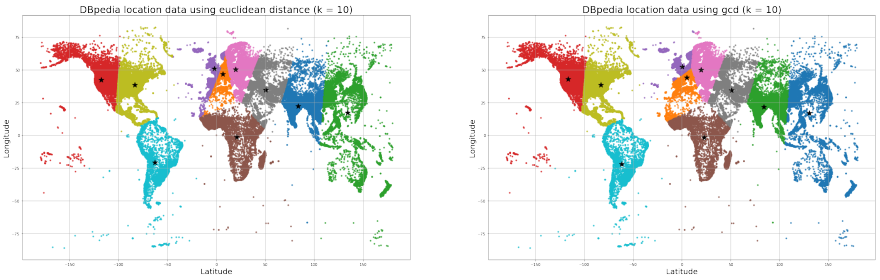
For $k = 2$ shown in Figure 5, the whole dataset was split into two clusters, one lied on America and another lied on the rest of the world. However, because of centroids in this dataset could represent the potential locations of the Wikipedia servers, only two centroids would be much less sufficient. Due to the distribution of the points, two distance algorithms again did not produce visually different results.



(a) euclidean distance (b) great circle distance

Figure 6: DBpedia location data (k = 6)

From Figure 6 above, we observed that with $k = 6$, the dataset was more separated than the previous outcome. However, the separation between different clusters seemed unnatural. It was due to the number of k was still not large enough to fully cluster the dataset. Thus, we summarized that for a large dataset, a minimum k was required to fully characterize the data.

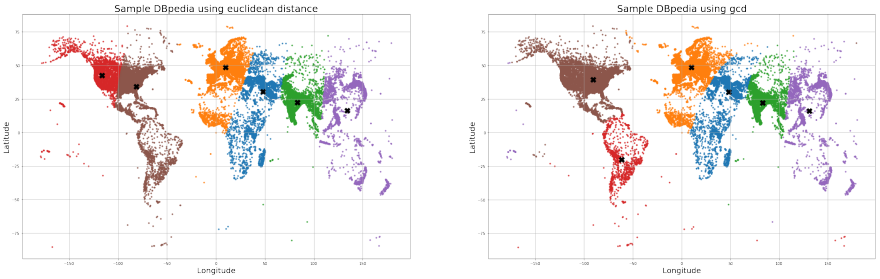


(a) euclidean distance (b) great circle distance

Figure 7: DBpedia location data (k = 10)

Finally for $k = 10$, Figure 7 showed us a better clustering results. With larger k, we were able to see separations between different continents for the first time. It was also the first time, we observed different results generated by the distance algorithms. Clusters produced with Euclidean distance algorithm, separate Europe vertically while the one with Great Circle distance separate Europe horizontally. Thus, we claimed that with a large dataset and large k, two distance functions would perform oppositely in high-density data groups.

In order to prove our claims, we randomly sampled 2% data points from DBpedia dataset, then did a k-mean clustering with $k = 6$ on both distance algorithms.



(a) euclidean distance (b) great circle distance

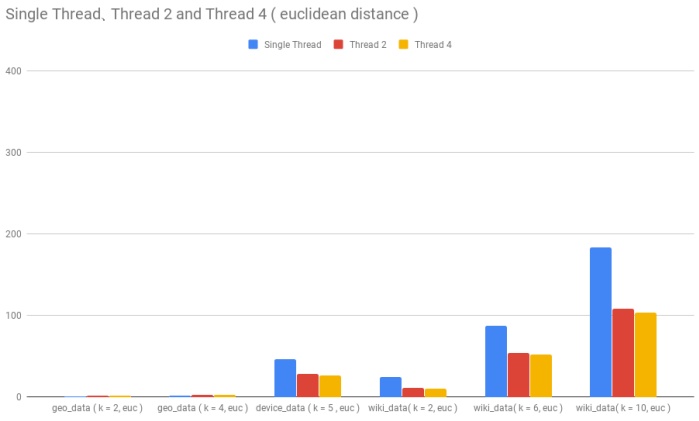
Figure 8: Sample DBpedia location data (k = 6)

From the result in Figure 8 we obtained, we noticed the two distance algorithms produced different this time. For Euclidean distance function, it separated America into two evenly. On the contrary, the Great Circle distance algorithm chose to divide North America into North and South. Thus, we could conclude that two distance algorithms would perform differently to each other when calculating k-means with large and sparsely distributed datasets.

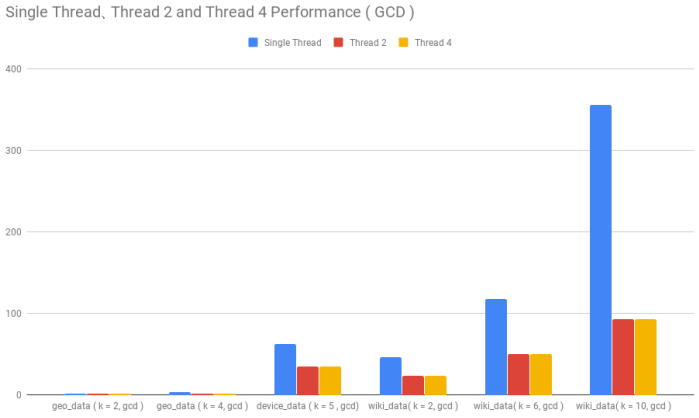
Runtime Analysis

After experimenting on all three datasets on the pseudo cluster, we found out that Spark performed differently on all three datasets. Therefore, we planned to find out how the size of a dataset, as well as two distance algorithm, would impact the Spark runtime performance. At the same time, we tended to find out how Spark scalability under multithreading.

To conduct this experiment, we ran our k-means clustering implementation on each dataset with different k for 10 times using a single thread, 2-threads, and 4-threads. Then we average the final wall time of each trail and compare them under the same thread, dataset and k conditions.



(a) euclidean distance

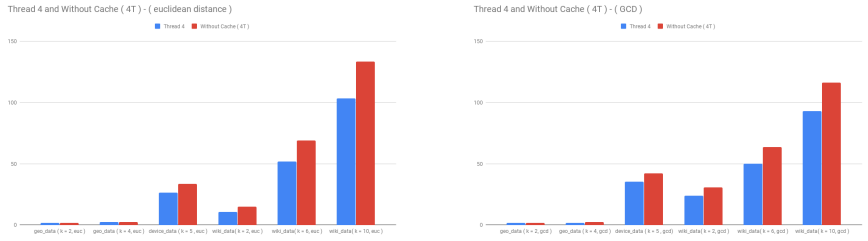


(b) great circle distance

Figure 9: Run time comparison

From the bar charts in Figure 9, we noticed that the scalability of Spark varied between various sizes of datasets. For a fairly small dataset like the synthetic location data, Spark did not scale under multithread at all. Single thread tended to perform similar or even better than the multithread. With a medium-sized dataset like device location data which contained around 280 thousand data points, Spark was able to shorten the time almost by half under 2-threads. For a large dataset such as DBpedia, Spark could save a significant amount of running time under multithread compared to single thread environment. However, Spark did not scale or barely scale when the number of threads was greater than two. It was also interesting to see that even though the Great Circle distance algorithm was more computational expensive compared to Euclidean distance algorithm, it performs slightly better with large dataset under multithreaded environment. For our DBpedia test, the Great Circle distance algorithm with multithread cost only one-fourth of the time used by a single thread and took less time than Euclidean distance.

Beside the above experiments, we also tested how persistent(cached) RDDs will affect the Spark performance.



(a) euclidean distance (b) great circle distance

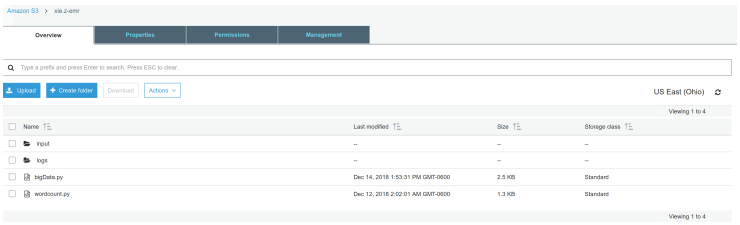
Figure 10: Cache comparison

From Figure 10, we could observe that the overall impacts on the performance were similar to our multithreaded experiments. Spark performed similarly when process small datasets. As the size of datasets increasing, the cached RDDs would have a better performance compared to without cached RDDs.

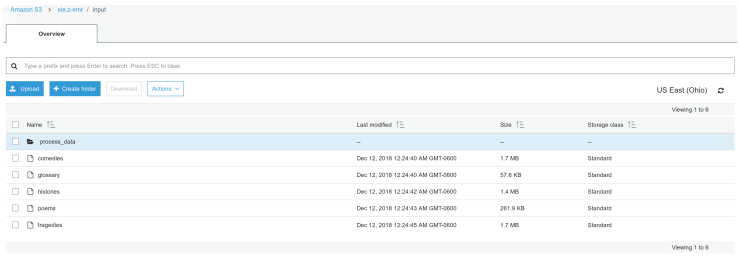
Big Data and Cloud Execution

After we finished our testing on provided datasets on the pseudo cluster, we then performed a k-means clustering on our own large-scale datasets. Our large-scale datasetsInstagram User check-ins with Geo pinpoints was obtained from kaggle.com. It was a dataset that contains Instagram users check-in geographical locations. It contained around 3.55 million observations with five attributes. They were number (not labeled), check-in time, check-in longitude, check-in latitude and hashed string (not labeled).

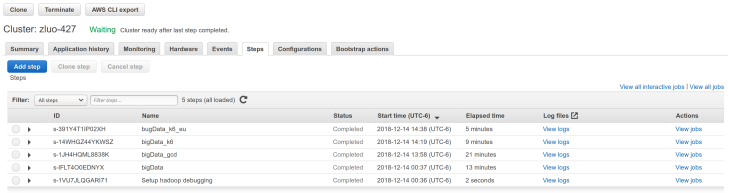
For our real cluster, we used Amazon Web Services as our platform. Specifically, Amazon EC2 as our computational instance and Amazon Simple Storage Service (S3) as our online storage system where input and output files for our Spark application is hosted. The cloud is configured with reference to the Lab11: Cloud Execution for Real instructions. We uploaded our preprocessed dataset and Spark python executable to S3. Then by creating a new cluster on Amazon EMR, we specified input and output destination as well as executable locations. The following Figure 11 shows how AWS was organized for our cloud execution.



(a) S3 Storage



(b) input folder for Spark



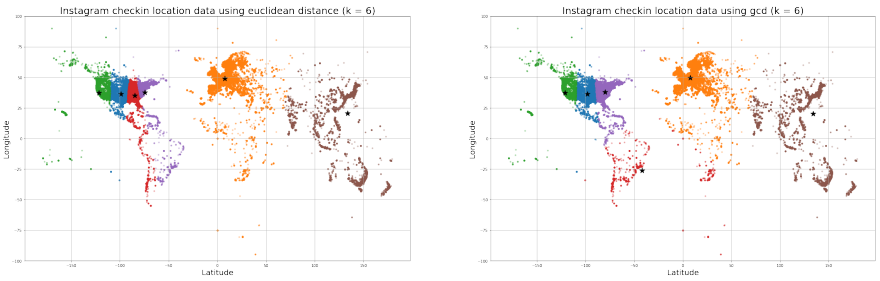
(c) EMR Jobs

Figure 11: AWS details

Since our cluster was running on AWS m4.large instance which had only one physical core, we were able to see the runtime result shown in Figure 11(c) that k-means clustering with Euclidean distance algorithm performed much better than using Great Circle distance algorithm as we shown in the previous section. Also, by comparing the running time between our DBpedia and Instagram data with $k = 10$, we noticed the time did not increase linearly. Our big dataset contained 7 times more data points to DBpedia, however then time only double. Thus, we could say, Spark tented to perform better when the size of the dataset

increased.

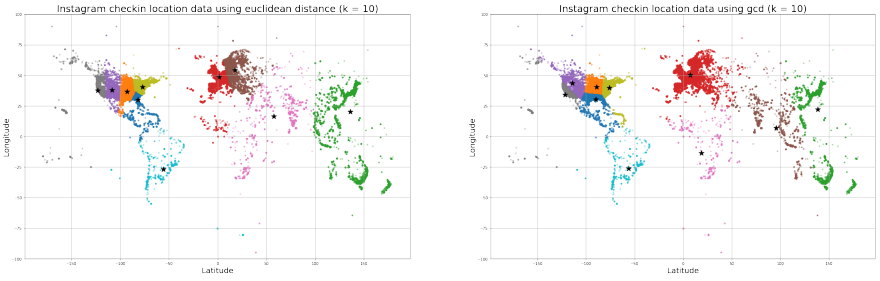
For our experiments, we tried $k = 6$ as well as $k = 10$ for our large dataset. As Figure 12 shown below, we observed that due to the distribution of our data points, the dataset was clustered really well. It successfully separated Europe and Asia. For America, from our previous conclusion, we could notice there were much more data points in North American compared to the rest of the world. Therefore, we would need more k in order to fully cluster our large dataset.



(a) euclidean distance (b) great circle distance

Figure 12: Instagram check-in location data (k = 6)

With $k = 10$ shown in Figure 13, we observed that our data was clustered much better than the previous one. Each continent was separated from each other. In North America, it was divided more clearly.



(a) euclidean distance (b) great circle distance

Figure 13: Instagram check-in location data (k = 10)

As for big data analysis for our dataset, with k-means clustering, we were able to understand the distribution of Instagram users. There were an only small amount of Instagram users in South America and African. Instagram users in Asia mostly lied on the west and south coast of China as well as east and west coasts of Australia. There were barely any users in the middle east and Russia. A large amount of Instagram users existed in Europe and majority

of them were from North America. Thus, we were able to predict what would be the potential next new market for Instagram.

Big Data Application

After analyzing the results, one potential application for clustering of Instagram check-in geo-location data is to combine with other data mining algorithm to form a recommendation system. That is, the Instagram posts can first be clustered based on the check-in locations, and whenever the users want to discover interesting posts, the system can recommend within-cluster posts to the users. Which posts within the cluster to recommend depends on the users' preference but the posts being inside the cluster ensures that the users are receiving the recommendations that are close to them geographically.

Conclusion

The k-means clustering problem computationally difficult; yet with simple heuristics, the algorithm quickly converges to a local optimum. The final output of the result is heavily influenced by the choice of a similarity measure and the number of the clusters. The conclusion of which similarity measure is better only depends on the problem itself. In the scenario of Instagram check-in location dataset clustering problem, great circle distance might be a more practical and hence better solution, since the earth is presumably spherical and the great circle distance makes better sense in measuring the distance between two points. In turns of the number of clusters, it can be determined computationally sometimes but not always. Sometimes the problem gives insights and intuitions; however, the determination of the number of clusters should avoid data snooping to ensure that our learning from the data is unbiased and feasible.

As for Spark and multi-thread execution, the importance and benefits of parallel computing are obvious especially when the dataset is large. While the runtime difference is insignificant on a small dataset, 2-thread execution takes about only half of the single-thread execution. Even though parallel computing improves the runtime significantly, the overall runtime does not improve proportionally to the increase in the parallelism. In our project, the difference between the runtimes of 2-thread execution and 4-thread execution is very small. Thus, we concluded that multithread was necessary for Spark performance, however, thread number larger than two would be wasted.

Lesson Learned

While the implementation of data mining algorithms such as k-means requires unambiguous correct decisions and actions, there might not be an absolute right answer in the field of big data application. The problems should be analyzed with reference to its context and conclusion should be drawn from the unbiased

data. In this project, a significant amount of the time is devoted to the discussion on which distance measure is better and how many numbers of the cluster is optimal. Clearly, there does not exist a correct answer. Yet, through discussions and checking references, this project teaches an important lesson of approaching the problem from the problem's perspective.

Future Work

In the field of data mining, the k-means algorithm is often discussed together with the EM algorithm which stands for expectation and maximization. Some general idea for future work would be EM algorithm related, such as EM algorithm (Gaussian Mixture Modeling) implementation, efficiency and runtime comparison with k-means, and etc. Also, for Spark application, since we discovered the non-linear performance increased on increasing dataset, it would be interesting to find out what would be the upper bound for the performance increase. Fundamentally, the ultimate goal is to understand we can achieve in big data application and cloud computing, while the EM algorithm is just a relevant and specific topic to hit on.