A Comparison of Clustering Algorithms: CURE, DIANA, AGNES, BIRCH

Lucia Fuentes, Mazen Meziad, Adi Tangirala, Hanyang Zhang

I. INTRODUCTION

The motivation of the project is to focus on four different hierarchical clustering algorithms, and compare them using silhouette index based on scores and visualizations. Hierarchical clustering is applied based on the group similarities of the data. Hierarchical clustering is a type of clustering in which data sets are split into different clusters based on how similar the data points are from each other taking hierarchy into account. The four clustering algorithms that will be looked at are BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies), CURE (Clustering Using Representatives), DIANA (Divisive Analysis Clustering) and AGNES (Agglomerative Nesting). The goal of the investigation is to discover the differences and similarities between the four hierarchical clustering algorithms when applying real world and artificial data, so data scientists could have better understanding of when to use each of them and how although they are all algorithms for clustering, it is important to use specific algorithms to specific data sets to get more accurate results.

II. BIRCH

A. Literature review

There are three references that were used for researching the BIRCH algorithm. The first one is the paper that introduced the BIRCH algorithm in 1996. It introduces the algorithm, stating that they will be talking about data clustering and how their main goal was to try to find ways in which they could cluster large data sets in a more efficient way. This is how BIRCH came to be. They then go on to discuss past algorithms and their disadvantages and some advantages of BIRCH such as its use of local clustering, that it makes full use of available memory and the fact that it does not require the whole data set from the start to work. The paper also discusses the steps for computing BIRCH (there are four that will be discussed later in this paper) and the mathematics behind it. Overall, this paper does a very good job of convincing the reader why this algorithm works by providing proof not only its applications, but also showing the mathematics and time complexities and by showing how it is improving upon past algorithms. [1]

The next reference used was an article published by the University of California, Berkley were they discuss the different properties of BIRCH such as how it makes local decisions and how sparse regions are handled as outliers. The article also defines different definitions used for the distances such as the centroid, radius and D4 which are discussed in this paper later on. They discuss the flaws of BIRCH such as the fact that it has no bounds and that the performance analysis is unconvincing which is something that we would agree on, as it was really hard to find this information for the algorithm. Finally, they discuss why BIRCH is relevant which is how it performs very well with just a one time pass of the algorithm on the data by clustering and throwing out outliers. [2]

Finally, the last reference comes from our textbook "Data Mining Concepts and Techniques" and it is a section 10.3.3 which discusses the BIRCH algorithm. The book discusses BIRCH by showing how it does hierarchical clustering using CF trees in order to have very good speed and scalability when using large data sets which were two problems that past algorithms could not overcome. It also discusses the math that was discussed in the past references and goes on to describe the two most important steps in BIRCH that are not optional (described later on in this paper). [3]

B. Theoretical topic

Hierarchical clustering is a type of clustering in which data is split into different groups of objects that are alike. An algorithm that does this is the BIRCH algorithm, which stands for "Balanced Iterative Reducing and Clustering using Hierarchies". This algorithm was first invented in 1996 by Tian Zhang, Raghu Ramakrishnan and Miron Livny[1] in hopes of being able to cluster large amounts of data given a small number of resources. It was able to do this by using a multiphase clustering technique in which the first clustering scan already gives a good clustering result. Additional scans can further improve the quality. It was the first algorithm to be able to deal with data points that are not supposed to be in the primary pattern in an effective way.[2]

BIRCH was an important invention because it helps solve scalability issues and is able to undo what was done in previous steps: two issues that were prevalent in past clustering algorithms. Its time complexity in O(n), with n being the number of objects that need to be clustered and it works by only going through the data once in linear time for good results, and any additional number of times for better results.

C. BIRCH's Steps

BIRCH has four steps that must be followed for its completion:[3]

1) Create a CF-Tree based on the initial data set: It is important to keep the original clustering structure, while building the tree in an incremental, dynamic approach. This tree has to be balanced. The CF-Tree

is made up of nodes with pointers to child nodes and drdifferent clustering features, which are 3D vectors giving information about clusters of objects defined as: "

$$CF =< n, LS, SS >$$

[3] where LS is the linear sum of the n points and SS is the square sum of the data points. From here we can derive other statistical information for the given cluster, such as its diameter D, radius R, and the centroid $x0$ which help show the tightness of a cluster around the centroid, defined as follows, [3]

$$\boldsymbol{x_0} = \frac{\sum_{i=1}^{n} \boldsymbol{x_i}}{n} = \frac{LS}{n}$$

$$R = \sqrt{\frac{\sum_{i=1}^{n} (\boldsymbol{x_i} - \boldsymbol{x_0})^2}{n}} = \sqrt{\frac{nSS - 2LS^2 + nLS}{n^2}}$$

$$D = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} (\boldsymbol{x_i} - \boldsymbol{x_j})^2}{n(n-1)}} = \sqrt{\frac{2nSS - 2LS^2}{n(n-1)}}$$

[3]

Clustering features help make the information stored on each cluster minimal, as they only need a space size that is constant to store the clustering feature instead of allocating space for each detailed information. Thus creating the efficiency in space that is key for BIRCH.[3]

2) This phase is optional. It builds a smaller CF tree which includes the range that is needed for this specific data set. For phase 3 to be successful, it has a certain range of input sizes in which they perform the best when taking into consideration speed and quality. In order to ensure that the CF-tree created in Phase 1 is in this range, we perform Phase 2 in order to try to be within this range.
3) Global Clustering: clustering of the leaf nodes in the CF-tree in order to get rid of outliers and to group smaller groups of clusters into larger ones. It uses the distance metric D2, the average inter-cluster distance (shown in 1), or D4, the variance increase distance (shown in 2), which is calculated by the CF-tree's vector in order to be more efficient.

$$D2 = \left(\frac{\sum_{t=1}^{N_1} \sum_{t=N_1+1}^{N_1+N_2} (\overrightarrow{X_1} - \overrightarrow{X_2})^2}{N_1 N_2}\right) \frac{1}{2}$$

Fig. 1. Average Inter-Cluster Distance: D2 [1]

4) This phase is optional. Before this, the data set has only been scanned once, so by using phase 4, the data can be more accurate. This phase continues refining the CF tree and eliminating outliers from the data set in order to have a data set with better clusters. It does this by using the centroids that were produced in the prior phase as seeds. It then redistributes the data points by its closest seeds, thus obtaining a new cluster.

$$D4 = \sum_{k=1}^{N_1+N_2} (\overrightarrow{X_k} - \frac{\sum_{l=1}^{N_1+N_2} \overrightarrow{X_l}}{N_1+N_2})^2 - \sum_{t=1}^{N_1} (\overrightarrow{X_t} - \frac{\sum_{t=1}^{N_1+N_2} \overrightarrow{X_t}}{N_1})^2 - \sum_{j=N_1+1}^{N_1+N_2} (\overrightarrow{X_j} - \frac{\sum_{l=N_1+1}^{N_1+N_2} \overrightarrow{X_l}}{N_2})^2$$

Fig. 2. Variance Increase Distance: D4[1]

III. CURE

A. Literature Review

There were two sources used for the research of the CURE algorithm. The first source was the original paper that was published to introduce CURE and show why it was an important algorithm. Throughout the paper, the authors try to convince the reader that CURE is a good algorithm by first talking about some drawbacks of past clustering algorithms such as the fact that past clustering algorithms were not able to deal with large data sets, as they would break down. Then, it discusses the steps of CURE by first explaining it conceptually and then showing the pseudocode and talking more about it in those terms. This paper does a very good job of explaining the space and time complexities of the algorithm as further proof that it is a very good for clustering large data sets. Another thing that this paper does very well is the fact that they not only talk about how CURE is a very efficient algorithm and say that it is better than past algorithms, but they show proof as graphs comparing it to another very efficient algorithm (BIRCH) and showing that CURE's execution time is faster. [4]

The next source gives a brief explanation of how CURE works and why it is important. It starts by explaining hierarchical clustering and that it is used for forming spherical and non-spherical clusters. Then it explains the algorithm's steps for working, by showing graphics of its architecture, how the cluster creation works with BIRCH visually, and then explaining the two graphics with a series of steps needed in the procedure. [5]

B. Theoretical Topic

CURE is a clustering algorithm that stands for "Clustering Using Representatives". It was first proposed in a paper written by Sudipto Guha, Rajeev Rastogi and Kyuseok Shim that was published in 1998 as a collaboration between Stanford University, Bell Laboratories and Korea Advanced Institute of Science and Technology[4]. During the time when CURE was released, the clustering algorithms that were the most popular named hierarchical and partitioning clustering algorithms had multiple drawbacks such as having multiple limitations in respect to the shape and the density of the clusters being created and also problems with scalability.

CURE aimed to improve upon the hierarchical and partitioning clustering algorithms. It was specifically designed to improve the coverage of what shapes of clusters can be identified in a good space complexity meant for large data sets. CURE's time complexity for lower dimensions is $O(n^2)$ and for worst case in higher dimensions, it is $O(n^2 log(n)$. Its space complexity is $O(n)$.

CURE has the ability to cluster different shapes and covers outliers better than other algorithms which helps it produce clusters that have a higher quality than the clusters being produced with other clustering algorithms. It is meant to be used on large data sets that are specialized and non-spherical shaped clustering. Some of CURE's drawbacks are that its time complexity is very slow, but this can be mitigated, and that as the dimensions gets higher, the performance gets worse. However, it is able to scale large data sets while still having efficiency, which is an advantage of CURE because a lot of other algorithms cannot do this.

C. CURE's Steps

These are the steps to perform CURE and are also displayed on figure 5 and figure 6 below:

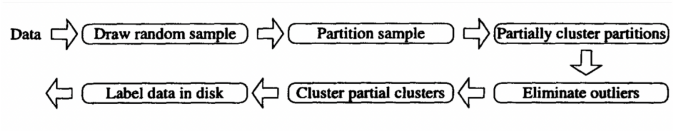


Fig. 3. Overview of CURE [4]

1) The algorithm starts by drawing a random sample of the data set and partitioning it into small cluster.
2) It then identifies multiple representatives in each cluster and moves them by alpha closer to the centroid of their cluster.
3) Pass
4) Find the shortest distance between the representatives and merge the clusters with the shortest distance between their representatives until the number of clusters wanted is reached
5) after every merge, pick new representatives and remove outliers by identifying clusters that are most slowly increasing in size and not merging.
6) Lastly, label all of the data by finding the closest representative to it and assigning it to that cluster.

The pseudo-code for the CURE algorithm works with two functions, one that creates clusters and another one that merges the clusters. The cluster algorithm takes as parameters the data set and the desired number of clusters to create the clusters that the user wants and it works as a for loop until the clusters needed have been created. The merge algorithm takes as a parameter two clusters, merges them and uses a k-d tree to know if there are any clusters that had one of the merged clusters be its closest cluster.

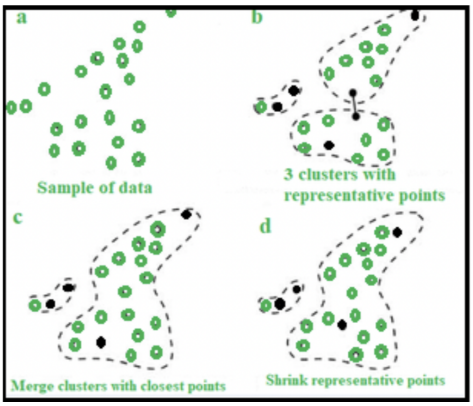


Fig. 4. Overview of CURE[5]

```
procedure merge(u, v)
begin
1.  w := u ∪ v
2.  w.mean := (|u|u.mean+|v|v.mean)/(|u|+|v|)
3.  tmpSet := ∅
4.  for i := 1 to c do {
5.      maxDist := 0
6.      foreach point p in cluster w do {
7.          if i = 1
8.              minDist := dist(p, w.mean)
9.          else
10.             minDist := min{dist(p, q) : q ∈ tmpSet}
11.         if (minDist ≥ maxDist){
12.             maxDist := minDist
13.             maxPoint := p
14.         }
15.     }
16.     tmpSet := tmpSet ∪ {maxPoint}
17. }
18. foreach point p in tmpSet do
19.     w.rep := w.rep ∪ {p + α*(w.mean-p) }
20. return w
end
```

Fig. 5. Algorithm to merge cluster

```
procedure cluster(S, k)
begin
1.  T := build_kd_tree(S)
2.  Q := build_heap(S)
3.  while size(Q) > k do {
4.      u := extract_min(Q)
5.      v := u.closest
6.      delete(Q, v)
7.      w := merge(u, v)
8.      delete_rep(T, u); delete_rep(T, v); insert_rep(T, w)
9.      w.closest := x /* x is an arbitrary cluster in Q */
10.     for each x ∈ Q do {
11.         if dist(w, x) < dist(w, w.closest)
12.             w.closest := x
13.         if x.closest is either u or v {
14.             if dist(x, x.closest) < dist(x, w)
15.                 x.closest := closest_cluster(T, x, dist(x, w))
16.             else
17.                 x.closest := w
18.             relocate(Q, x)
19.         }
20.         else if dist(x, x.closest) > dist(x, w) {
21.             x.closest := w
22.             relocate(Q, x)
23.         }
24.     }
25.     insert(Q, w)
26. }
end
```

Fig. 6. Algorithm to create clusters

IV. DIANA

A. Literature Review

The first reference is called Finding Groups In Data: An introduction to Cluster Analysis written by Leonard Kaufman and Peter J. Rousseeuw. The book introduces the concepts of clusters and different programs applications to clusters such as Partitioning Around Medoids (PAM), Clustering Large Applications (CLARA), Fuzzy Analysis (Program FANNY), and most importantly Agglomerative Nesting (AGNES) and Divisive Analysis (DIANA).

In the DIANA section of the book, the author first shows the general concept of DIANA and stated why divisive method has been ignored. He stated: "Most books on cluster analysis pay little attention to divisive techniques, and many software packages do not include divisive algorithms at all (P. 253)." Like he said, popular programming languages such as python does not have package includes DIANA algorithms despite its packages abundance compare to other programming languages. Because DIANA recursively splits data points into two more clusters, the algorithm contains $2^(n-1)-1$ possibilities, where n is the number of objects. It is the the reverse of AGNES where it has n(n-1)/2 combinations. Then the author shows how to use the algorithms in real world dataset and deep dive into the DIANA step by step.

The second reference is called Divisive Analysis (DIANA) of hierarchical clustering and GPS data for level of service criteria of urban streets published by Ashish Kumar Patnaik, Prasanta Kumar Bhuyan, and K.V. Krishna Rao. The article introduces a real world problem of using DIANA to classify urban road networks in India into number of streets based on categories. DIANA clustering could be used based on a GPS receiver. They apply distance matrix into DIANA algorithms and find out the minimum distances between destinations which provide results and conclusion. Finally, the authors use results they calculated to form a transportation map in city of Mumbai and how the measures for number of clusters using DIANA. In conclusion, the authors successfully apply DIANA algorithm into real world problem efficiently. They stated: " It has been observed that DIANA is a very successful clustering tool that can be applied for all kinds of urban roads have varying traffic flow." From their study, DIANA algorithm is very efficient when applying urban and traffic network applications, which could contribute to city transportation planning, road network design, and traffic facilities planning in general.

B. Theoretical Topic

Diana is a clustering algorithm that stands for "Divisive Analysis Clustering" is part of a group of clustering algorithms called "hierarchical clustering"[7]. Hierarchical clustering is a type of clustering analysis whose main goal is to be able to build clusters with hierarchies. There are two types of hierarchical clustering: agglomerative clustering which is a "bottom-up" approach and divisive clustering which is a "top-down" approach.

DIANA corresponds to the divisive clustering category because the way that DIANA's algorithm works is that it puts all of the data in the same cluster and then starts splitting the big cluster into smaller ones by half until the data has been clustered correctly. This algorithm is unique compared to other hierarchical clustering algorithms because most of the other ones use agglomerative clustering rather than divisive [8].

Popular programming languages such as python does not have package includes DIANA algorithms despite its packages abundance compare to other programming languages. Because DIANA recursively splits data points into two more clusters, the algorithm contains $2^(n-1)-1$possibilities, where n is the number of objects. It is the the reverse of AGNES, a popular agglomerative clustering algorithm, that has $n(n-1)/2$ combinations. [7]

C. DIANA's Steps

DIANA algorithm is the reverse version of agglomerative algorithms, which was introduced in 1990 by Kaufman and Rousseeuw. Based on the first reference written by them, the steps of DIANA is shown as follows: While DIANA forms one single cluster and split into two recursively, the first step of the algorithm is to put all data points into one cluster, and iterate until all the data points are assigned. In the description where the author divide cluster R into two clusters A and B, they use average dissimilarity to check every object in cluster A by the following formula:

$$d(i, A \setminus \{i\}) = \frac{1}{|A|-1} \sum_{\substack{j \in A \\ j \neq i}} d(i, j)$$

It computes the average dissimilarities of all the objects in cluster A, and recall that DIANA splits the cluster into two least similar smaller clusters. So the maximum object in cluster A will be moved, and then it will generate the new A and B clusters. The maximum object in cluster A will be remove and it goes to the cluster B which is the least similar cluster to A. The formula shown as follows:

$$A_{\text{new}} = A_{\text{old}} \setminus \{i'\}$$
$$B_{\text{new}} = B_{\text{old}} \cup \{i'\}$$

Then, after removing the maximum object in A and move it to b, check if other objects in cluster A could move to B where cluster a has at least two objects.

$$d(i, A \setminus \{i\}) - d(i, B) = \frac{1}{|A|-1} \sum_{\substack{j \in A \\ j \neq i}} d(i, j) - \frac{1}{|B|} \sum_{h \in B} d(i, h)$$

For this part of the algorithm, determine all the objects in cluster A and keep finding the maximum values and move them from cluster A to cluster B, and keep doing this procedure recursively until there is only one object in cluster A or the maximum value of dissimilarities is negative or zero. That is the signal of completing the first two clusterings A and B from R. And the steps goes on until each of the objects becomes a cluster.

Note: The diameter formula could be used to decide which cluster to split in DIANA, and it shows as follows:

$$\text{diam}(Q) = \max_{\substack{j \in Q \\ h \in Q}} d(j, h)$$

In order to explain the method, note that the example variable above contains a cluster Q splits into two smaller clusters j and h, where they belong to Q. Choose the largest to cluster and continue the DIANA procedure. So this is an overview of the theoretical aspect of DIANA algorithms, which could also be applied in python code or R package. The dendrogram is the graphical aspect of the algorithm, which has the same logic and procedure.

V. AGNES

A. Literature Review

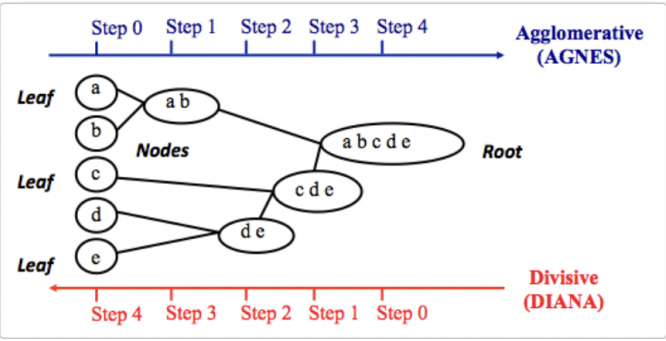
There were two references that were used fro AGNES. The first one was a website that talked about agglomerative hierarchical clustering and it was very helpful to understand

this type of clustering because it gave good background information on what hierarchical clustering was as a whole and then talked specifically about AGNES and how it used the "bottom-up" approach to computer its algorithm. It shows in a graph how DIANA and AGNES are the opposite of each other by showing how they handle clustering and then talks about the steps needed to compute the algorithm. Lastly, the article shows how to code the algorithm in the R programming language.[9]

The next reference used was an article published by the University of Cambridge on agglomerative clustering. They start the article by defining hierarchical clustering and the two types of approaches top-down or bottom-up, followed by starting to talk about agglomerative clustering and stating that it is the most widely used algorithm out of the two. The article then goes to extensively talk about dendrograms and how agglomerative clustering uses them to visualize their results. Finally, it talks about the steps and the mathematics of the way that the agglomerative algorithm merges its clusters. This article does a very good job at explaining the importance of dendrograms when visualizing AGNES that I had not realized when reading other articles like it because although other articles mention the use of dendrograms not in a way that makes them seem important. Also, the use of mathematics to explain the merging of the clusters was very helpful to understand AGNES further.[10]

B. Theoretical Topic

AGNES is a type of hierarchical clustering that stands for "Agglomerative Nesting". How AGNES works is that it is a "bottom-up" approach to clustering meaning that the algorithm starts by making each data point be its own cluster, and then starts continually merging them together until the right number of clusters has been achieved. At the end of the AGNES algorithm, the result is a tree-based representation called a dendrogram. The opposite of agglomerative clustering is the less widely used divisive clustering which was talked about above with DIANA, as its approach is "top-down". [9]



[9]

The first published reference of AGNES was made in 1964 by Ward. Agglomerative clustering is an algorithm that is widely used because it can result in useful taxonomic divisions between data points and because some implementations don't assume number of clusters, which makes it easier for the user to input however many they want. However, some drawbacks are that it can't take back cuts after they're made and its time complexity which is super low for large data sets as the number of variables increases, the performance of AGNES decreases.

C. AGNES' steps

Looking at the pseudo code below, we can follow the implementation of AGNES' steps. The algorithm below is pretty inefficient as its time complexity is $O(n^2 log(n))$, but it is easy to understand.

1) Compute NxN similarity matrix C.
2) Merge most similar clusters $n-1$ and then storing all of the merges in A.
3) SIM function compares similarity of clusters j with the merge of i and m.

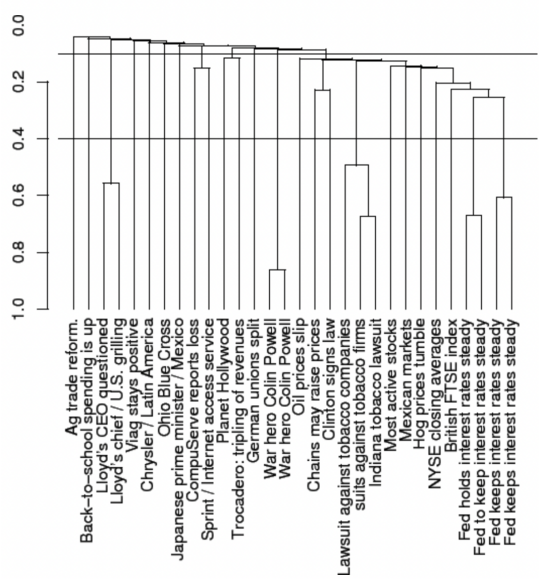
```
SIMPLEHAC(d_1, ..., d_N)
 1  for n ← 1 to N
 2  do for i ← 1 to N
 3     do C[n][i] ← SIM(d_n, d_i)
 4     I[n] ← 1  (keeps track of active clusters)
 5  A ← []  (assembles clustering as a sequence of merges)
 6  for k ← 1 to N − 1
 7  do ⟨i, m⟩ ← arg max_{⟨i,m⟩:i≠m∧I[i]=1∧I[m]=1} C[i][m]
 8     A.APPEND(⟨i, m⟩)  (store merge)
 9     for j ← 1 to N
10     do C[i][j] ← SIM(i, m, j)
11        C[j][i] ← SIM(i, m, j)
12     I[m] ← 0  (deactivate cluster)
13  return A
```

Fig. 7. Pseudo-code of AGNES Clustering [10]

When computing step 2 in AGNES, an assumption that is made about the merge operation is that it is monotonic.

Monotonic = "if $s_1, s_2, ..., s_{k-1}$ are the combination similarities of the successive merges of an HAC, then $s_1 >= s_2 >= ... >= s_{k-1}$ hold. A non-monotonic hierarchical clustering contains at least one inversion $s_i < s_{i+1}$ and contradicts the fundamental assumption that we chose the best merge available at each step." [10]

AGNES clustering is usually displayed as a dendrogram, as shown in figure 7. Every time that there is a merged in the algorithm, the dendogram represents it as a horizontal line. The dendrogram works by looking at it in a bottom-down way, we can see all of the merges that occurred historically within the clusters. [10] Dendrograms are really good for showing the groups of clusters that are formed within the dendrogram step-by-step.



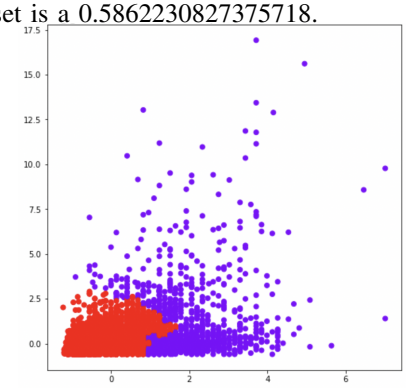
VI. Data Application

In order to show our data application, we used two ways to measure how each algorithm did on each data set: a visual representation on a scatter plot and the silhouette score. The silhouette score is a method of measuring the quality of clustering in a data set. For the silhouette score, the best score that someone can get is a 1 and the worst is a -1 which will be an important thing to note when looking at the silhouette scores that the different algorithms will be obtaining on the data sets. Also, all of our data sets have only numeric values as our algorithms only work with these type of values, there can be no characters or strings on the columns.

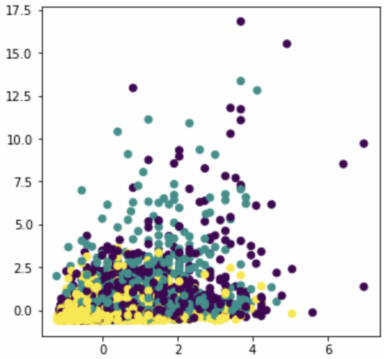
A. Credit Card Information

This data set has information on different people's credit card transactions based on their customer ID. It had information about people's account balances, purchases, how many purchases they made, the credit limit, and even the tenure, among others. However, the columns that we will be using when looking at our data and clustering it will be credit limit and payments and setting the cluster number to 3 for all of the algorithms. This is an artificially generated data set and it contains 8950 rows. [11]

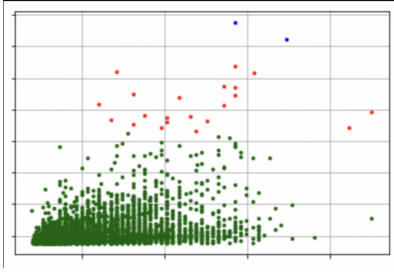
1) AGNES Algorithm on Credit Card Data Set: The silhouette score for this algorithm on the credit card data set is a 0.5862230827375718.



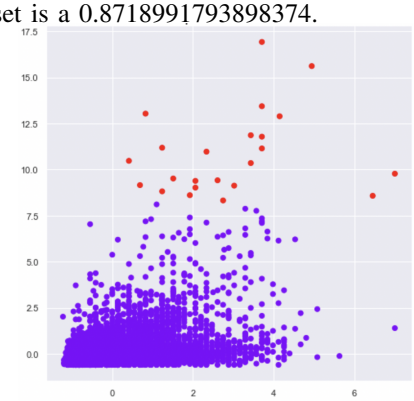
2) DIANA Algorithm on Credit Card Data Set: The silhouette score for this algorithm on the credit card data set is a 0.17845266791386857.



3) CURE Algorithm on Credit Card Data Set: credit card data set is a 0.9628260835987251.



4) BIRCH Algorithm on Credit Card Data Set: The silhouette score for this algorithm on the credit card data set is a 0.8718991793898374.



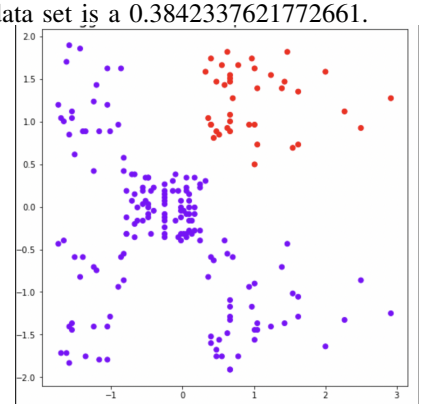
5) Analysis: After analyzing the data on all four of the algorithms, it seems that the algorithm that was able to cluster the data set the best was CURE, as its silhouette score was the highest at 0.963 and then BIRCH with a 0.872. AGNES and DIANA had significantly worse scores of 0.586 and 0.178 respectively. Visually, these results are very apparent because BIRCH and CURE both look very similar visually with the bulk of the data on the bottom left being one cluster and some outliers on the top right being the second cluster. It is also obvious that DIANA did not do a good job clustering because it is impossible to be able to determine where one cluster starts and where one ends. It is distinguishable with AGNES, but the result visually does not look correct. Based on this specific data set and the fact that it has so many rows of information (8950), it makes sense that the two algorithms that performed the best were BIRCH and CURE because they are made to work best with large

data sets.

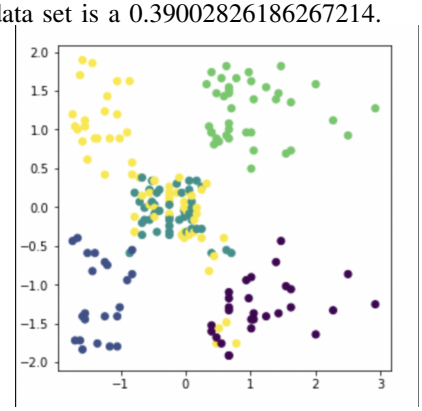
B. Mall Customers

This data set has columns on different data on mall customers such as their customer id, gender, age, annual income and how much money they spent on the mall. When performing the algorithms on using these data sets,the columns that we used for the mall customer data set were annual income and spending score and we clustered each clustering algorithm based on 5 clusters. This is an artificially generated data set and it contains 200 rows. [12]

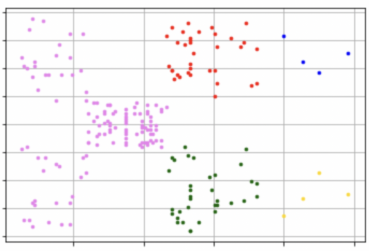
1) AGNES Algorithm on Mall Customers Data Set: The silhouette score for this algorithm on the Mall Customers data set is a 0.3842337621772661.



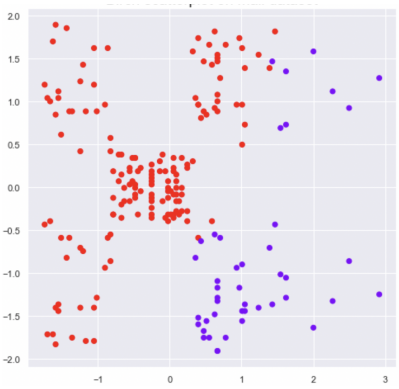
2) DIANA Algorithm on Mall Customers Data Set: The silhouette score for this algorithm on the Australian weather data set is a 0.39002826186267214.



3) CURE Algorithm on Mall Customers Data Set: The silhouette score for this algorithm on the Australian weather data set is a 0.5373456250675044.



4) BIRCH Algorithm on Mall Customers Data Set: The silhouette score for this algorithm on the Australian weather data set is a 0.392499029691066.

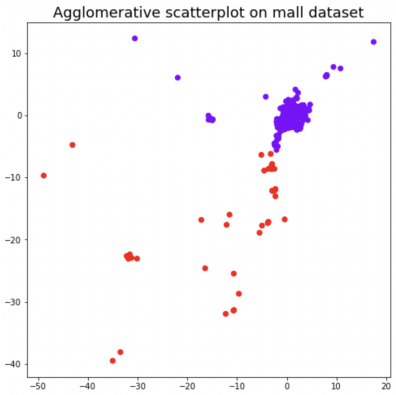


5) Analysis: After looking at the results from the credit card data set, all of the silhouette scores for BIRCH, DIANA and AGNES are very similar to each other with all of them ranging from 0.38-0.39, which is not a very good score. Looking at the results visually, although they were supposed to form a total of 5 clusters, it seems like AGNES and BIRCH formed only 2 which means that the algorithms were not clustering correctly. DIANA was able to show 5 colors in the graph, but they are mixed together so it is not easy to distinguish where some clusters end and where some begin. When looking at CURE, although its silhouette score was also not very good being 0.537, it was also still able to form five clusters that are not really mixed together but they are also not correct. As the mall customer data set only contains 200 rows, it makes sense that CURE and BIRCH weren't able to cluster it because they perform best with large data sets. It also makes sense that DIANA was able to show five colors unlike BIRCH and AGNES, as it is good with small data sets. However, the fact that it did not cluster correctly and the points seem to be mixed together shows that it is not very efficient. In conclusion, for the mall data set CURE performed the best again even though it was a smaller data set.

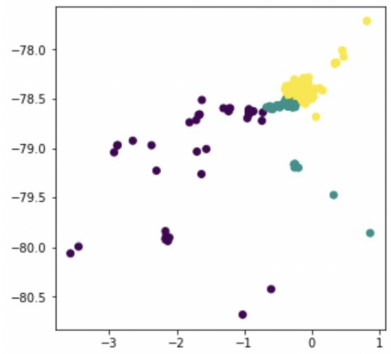
C. Taxi Routes

This data set has information on taxi routes in Quito and it has columns on pickup daytime, drop off daytime, latitude, longitude and trip duration, among other columns. The columns that we used in order to cluster were drop off latitude and drop off longitude and we clustered each algorithm on 3 clusters. This data set is a real-world data set and its information was acquired from the EC Taximeter app. The data set contains 90.3k columns. [13]

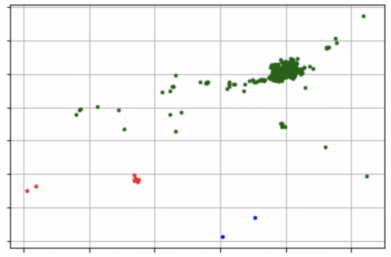
1) AGNES Algorithm on Taxi Routes Data Set: The silhouette score for this algorithm on the Quito taxi routes data set is a 0.9491396902162438.



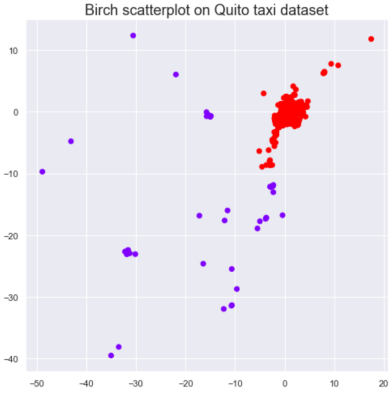
2) DIANA Algorithm on Taxi Routes Data Set: The silhouette score for this algorithm on the Quito taxi routes data set is a 0.5148876346484501.



3) CURE Algorithm on Taxi Routes Data Set: The silhouette score for this algorithm on the Quito taxi routes data set is a 0.9980404528229032.



4) BIRCH Algorithm on Taxi Routes Data Set: The silhouette score for this algorithm on the Quito taxi routes data set is a 0.9545593597058388.

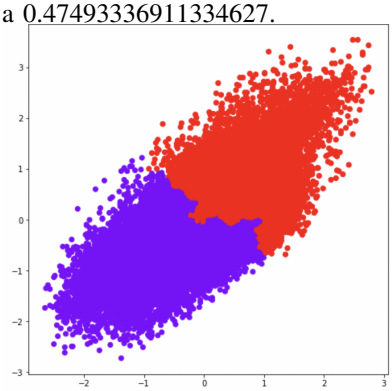


5) Analysis: When looking at the results for the taxi routes in Quito data set when applied to the four algorithms, CURE, AGNES and BIRCH had extremely good results for clustering as they had a range of .94-.99 silhouette scores with CURE having the highest score of .998. DIANA had the lesser score, of 0.515. Visually, the only algorithms that showed 3 cluster colors were CURE and DIANA and the colors are not really divided by clusters and I would even go as far as to say that DIANA seems to visually have the best clustering even though it is the one with the lowest silhouette score. Also, for this data set the there algorithms had slightly different looking data when plotted in the scatter plot with BIRCH being the one that is the most different. For this data set taking into account that CURE was able to show 3 colors and sort of clusters and it had the highest silhouette score, it is safe to assume that it was the one that could cluster the best taking into account the big number of rows of data that are getting clustered.

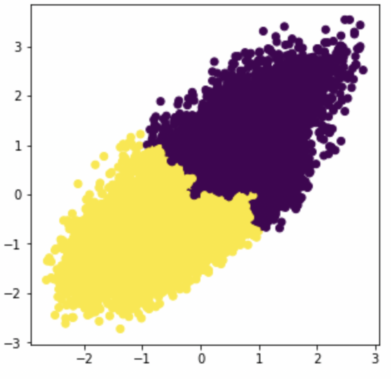
D. Rain in Australia

This data set has data on the weather in different cities in Australia and gives specific information about the weather such as the date, the specific city that it happened at, the minimum and maximum temperature, if there was wind, if it was humid, if it rained, among others. The two columns that we decided to perform our clustering were minimum and maximum temperature on 2 clusters. This data set is a real-world data set and its information was acquired from various weather stations in Australia over a 10 year period. The data set contains 145460 rows.[13]

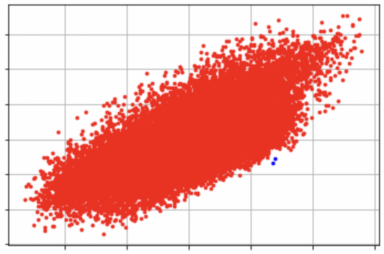
1) AGNES Algorithm on Weather Data Set: The silhouette score for this algorithm on the Australian weather data set is a 0.47493336911334627.



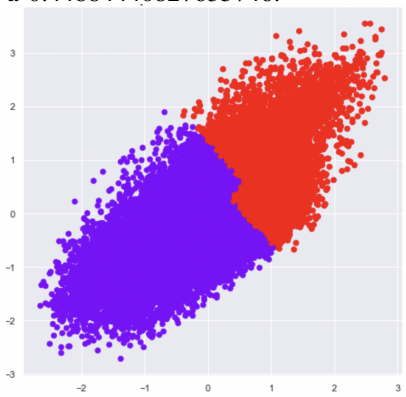
2) DIANA Algorithm on Weather Data Set: The silhouette score for this algorithm on the Australian weather data set is a 0.47493336911334627.



3) CURE Algorithm on Weather Data Set: The silhouette score for this algorithm on the Australian weather data set is a -0.03981961443192002



4) BIRCH Algorithm on Weather Data Set: The silhouette score for this algorithm on the Australian weather data set is a 0.44884440827635746.

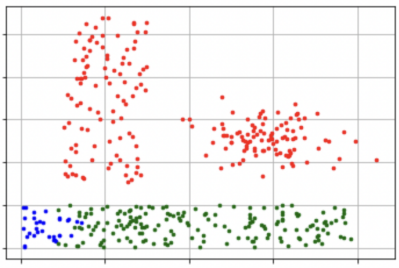


5) Analysis: Looking at the results of the weather data set being applied to the 4 algorithms and plotting the results as well as getting the silhouette score, BIRCH, DIANA and AGNES did the best. For the silhouette store, AGNES and DIANA had the higher scores of 0.4749 and the next algorithm that did the best with a score of 0.44884. These results are also apparent in the scatter plots because there are two colors showing and they all seem to cut the figure almost exactly in half. Surprisingly for this data set CURE is the one that is doing the worst with a negative score which had not not happened before. Although visually CURE did have two colors, the second color only shows up in like one dot, so it did very bad. Overall, the two algorithms that performed the best in this data set were AGNES and DIANA as they got the same silhouette score and their graphs look the same visually. This result makes sense as they are basically the same algorithm performing their steps on the opposite direction from each other.

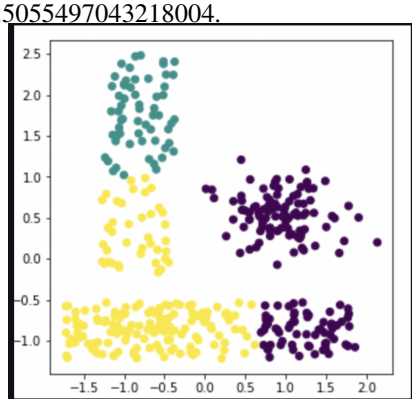
E. Built-in PyClustering Sample Data Set

This data set is built in in the "PyClustering" package and it originally demonstrates the functionality of the CURE algorithm. It is artificially generated so the numbers are random numbers generated by the package and we are using 3 clusters to cluster this data set. It has 400 rows of data.

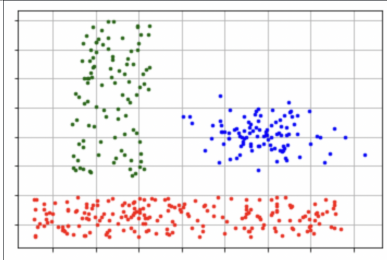
1) AGNES Algorithm on Built-in PyClustering Sample Data Set: The silhouette score for this algorithm on the Built-in PyClustering Sample Data set is a 0.3579676304220543.



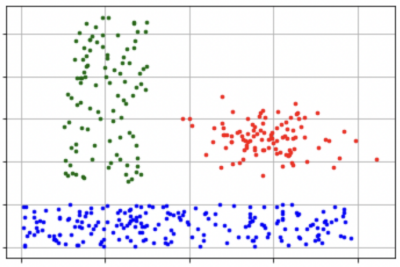
2) DIANA Algorithm on Built-in PyClustering Sample Data Set: The silhouette score for this algorithm on the Built-in PyClustering Sample Data set is a 0.5055497043218004.



3) CURE Algorithm on Built-in PyClustering Sample Data Set: The silhouette score for this algorithm on the Built-in PyClustering Sample Data set is a 0.5326039775548848.



4) BIRCH Algorithm on Built-in PyClustering Sample Data Set: The silhouette score for this algorithm on the Built-in PyClustering Sample Data set is a 0.6074953507066984.



5) Analysis: Looking at the silhouette scores of the results from performing the algorithms on the built-in data set, BIRCH was the algorithm that performed the best with a score of 0.607 and then it was CURE with a 0.533 score. Although their silhouette scores are different, visually, these two algorithms look exactly the same giving three clusters each of a different color. The clustering performed with this data set was really good. The other two algorithms weren't

able to cluster correctly, but they did both show 3 cluster colors which is good as the algorithms were not able to achieve this with other data sets.

VII. Results

Looking at the theory behind all of the different clustering algorithms and their applications on different real-life and artificially generated data sets, it seems as though the algorithm that works the best algorithm. CURE performed the best for 75 percent of the data sets both visually and with the highest silhouette score. It also seemed like the best algorithm theoretically because it was created in order to improve upon old algorithms with speed and its ability to be able to cluster large data sets. The only algorithm that seems to do almost as good as CURE is BIRCH, but as it was mentioned above, when they compared BIRCH and CURE in the original paper CURE did better. This theory was also proven in our results. The average for AGNES taking into account all of the data sets is 0.5687. For DIANA it is 0.4685, CURE it is 0.5982 and for BIRCH is 0.6551. Overall, the one that performed the best is CURE taking into account the averages and the theory.

VIII. Division of Labor

BIRCH Research	Lucia Fuentes
AGNES Research	Adi Tangirala
CURE Research	Mazen Meziad
DIANA Research	Hanyang Zhang
BIRCH Coding	Adi Tangirala and Lucia Fuentes and Mazen Meziad
AGNES Coding	Adi Tangirala and Mazen Meziad
CURE Coding	Mazen Meziad and Adi Tangirala
DIANA Coding	Hanyang Zhang and Mazen Meziad and Adi Tangirala
Final Essay	Lucia Fuentes
DIANA Writing Essay	Hanyang Zhang and Lucia Fuentes
Presentation	Adi Tangirala, Lucia Fuentes, Hanyang Zhang, Mazen Meziad

References

[1] Tian Zhang and Usama Fayyad. BIRCH: a new data clustering algorithm and its applications. Data Min. Knowl. Disc, pages 141–182, 1997.

[2] Fox , Armando, and Steve Gribble. “BIRCH: An Efficient Data Clustering Method for Very Large Databases.” Http://People.eecs.berkeley.edu, University of California, Berkley

[3] Han, Jiawei, et al. Chapter 10.3.3. Data Mining Concepts and Techniques, Morgan Kaufmann, San Francisco, Calif, 2011.

[4] Guha, Sudipto, Rajeev Rastogi, and Kyuseok Shim. "Cure: an efficient clustering algorithm for large databases." Information systems 26.1 (2001): 35-58.

[5] “Basic Understanding of CURE Algorithm.” GeeksforGeeks, 31 Aug. 2021

[6] Patnaik, Ashish Kumar, et al. “DIVISIVE ANALYSIS (DIANA) OF HIERARCHICAL CLUSTERING AND GPS DATA FOR LEVEL OF SERVICE CRITERIA OF URBAN STREETS.” Cyber Leninka, Association "Open Science", 29 Dec. 2016.

[7] Kaufman, Leonard, and Peter J. Rousseeuw. “Chapter 6: Divisive Analysis (Program DIANA).” Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, New York, 2005.

[8] Struyf, Anja, Mia Hubert, and Peter Rousseeuw. "Clustering in an object-oriented environment." Journal of Statistical Software 1.4 (1997): 1-30.

[9] Jovana, et al. “Agglomerative Hierarchical Clustering.” Datanovia, 20 Oct. 2018

[10] “Hierarchical Agglomerative Clustering .” Hierarchical Agglomerative Clustering, Cambridge University Press, 2008

[11] Bhasin, Arjun. “Credit Card Dataset for Clustering.” Kaggle, 2 Mar. 2018, https://www.kaggle.com/arjunbhasin2013/ccdata.

[12] Roshansharma. “Mall Customers Clustering Analysis.” Kaggle, Kaggle, 1 Nov. 2020, https://www.kaggle.com/mnavas/taxi-routes-for-mexico-city-and-quito

[13] Navas, Mario. “Taxi Routes of Mexico City, Quito and More.” Kaggle, 2 Aug. 2017, https://www.kaggle.com/mnavas/taxi-routes-for-mexico-city-and-quito.

[14] Young, Joe. “Rain in Australia.” Kaggle, 11 Dec. 2020, https://www.kaggle.com/jsphyg/weather-dataset-rattle-package.