SAINT MARY'S COLLEGE OF CALIFORNIA

DEPARTMENT OF MATHEMATICS

SENIOR ESSAY

Data Clustering Algorithms

Mentor:

Dr. Charles HAMAKER

Author:

Bryce CLOKE

Supervisor:

Dr. Kathryn PORTER

May 22, 2017

Abstract:

Data clustering algorithms are tools used in Cluster Analysis to quickly sort and identify groups of data points such that each point in these groups or "clusters" is similar in some way to the other points in the same cluster. Data clustering is a very broad field that has many different applications and corresponding algorithms. While there are a plethora of data clustering algorithms, this paper will look in-depth at the k-means sorting algorithm and a variation, the CLINK hierarchical sorting algorithm, and DBSCAN. The pros and cons, efficiency, applications, and possible improvements of these algorithms will be discussed, as well as their major differences.

1 Introduction

1.1 Background

Data clustering algorithms are part of a larger subject called **Cluster Analysis**. Cluster analysis itself is simply defined as the study of methods that divide data sets into groups (clusters) that are useful in some arbitrary way. A simple example might be sorting a box of Legos. The pieces (or objects) have different shapes, colors, and/or sizes. Let's say we want to sort the Legos by color, but how would one go about doing that? You might tell me to start by picking up a Lego, which happens to be blue, and put it aside in a pile corresponding to its color, repeating until there are no more objects left to sort. Well what you just described is a data clustering algorithm! The "clusters" in this example would be the different color piles, and the data points would be the pieces themselves. And while the above-stated algorithm might work well for a small number, what if we had billions of Legos and a thousand colors? What about one-hundred thousand colors? What about one-hundred billion? What about sorting by other additional features, like size? The means used to accomplish such a spectacular, sorting feat are specialized data clustering algorithms.

1.2 Definitions

- A **Data Point** is an object (sometimes called an **observation**) which has various properties associated with it.
- A **Dataset** is a collection (usually called a set) of data points.
- A **Sorting Algorithm** is a mathematical method that is applied to a dataset to divide the data into clusters.
- A **Cluster** is a grouping of data points which share some arbitrarily defined property, usually created via a sorting algorithm.

1.3 Dissimilarity and Similarity Measures

When discussing how data in a data set are similar or dissimilar, we will be referring to their appropriate distance function (for most sorting algorithms), using a distance function D. A distance function can be defined as follows:

A **distance function** D on a dataset X is a binary real-valued function that satisfies the following conditions:

- $D(x, y) \geq 0$ (Nonnegativity);
- $D(x, y) = D(y, x)$ (Symmetry or Commutativity);
- $D(x, y) = 0$ iff $x = y$ (Reflexivity);
- $D(x, y) \leq D(x, z) + D(z, y)$ (Triangle Inequality);

where x, y, z are arbitrary data points in X (Anderberg).

If our distance function satisfies all four above properties, we can also call it a **distance metric**.

1.4 Stirling Numbers of the Second Kind

Stirling numbers of the second kind count the number of ways we can partition a set of N objects into K non-empty subsets. They are found in the following formula:

$$S(N, K) = \frac{1}{K!} \sum_{i=0}^{K} (-1)^{K-i} \binom{K}{i} i^N,$$

and can be approximated as $\frac{K^N}{K!}$.

1.5 Types of Sorting Algorithms

Sorting algorithms fall under several types of classifications, including: **Centroid-based clustering**, **Connectivity-based clustering (hierarchical clustering)**, **Density-based clustering**, and other classifications. The scope of this paper will focus on an in-depth overview of each of the above classifications mentioned by name: a definition, an example, and a discussion. These classifications are based on how the algorithms are constructed and on what kinds of datasets they are used on. One kind of algorithm might be more effective than another for a multitude of reasons, as we will explore later on. The first type of algorithm that we will discuss is the **K-Means Clustering Algorithm**, which is a kind of centroid-based clustering algorithm.

2 Centroid-Based Clustering

Centroid-based clustering algorithms rely on the use of identifying and manipulating either true or pseudo **centroids**. Centroids are defined as being the center of mass of a geometric object of uniform density. The k-means algorithm seeks to first create k pseudo-centroids from the original dataset and then manipulate their position to locate the true centroid of the clusters within the dataset.

2.1 K-Means Clustering Algorithm

K-means clustering is a very popular, centroid-based sorting method used in data mining and machine learning, and can be seen as a foundation or starting point for other sorting algorithms. K-means clustering aims to partition n objects into k clusters, where each object is associated with the closest of k centroids, acting as a temporary pseudo-cluster until the algorithm completes. Following this algorithm to completion partitions the data into k subsets of the original dataset.

2.1.1 The K-Means Algorithm

Given an initial set T_1 consisting of k centroids $\{m_1^{(1)}$,...,$m_k^{(1)}\}$, we follow a repeating, two-step process to cluster the data.

Assignment:

Assign each object x_p to the cluster $C_i^{(t)}$ whose centroid $m_i^{(t)}$ has the smallest euclidean distance, such that:

$$C_i^{(t)} = \{x_p : d(x_p, m_i^{(t)}) \leq d(x_p, m_j^{(t)}) \ \ \forall j, 1 \leq j \leq k\}$$

where d is the Euclidean distance between the two points and each object x_p in the dataset is assigned to **exactly one** cluster $C_i^{(t)}$.

Update Step:

Next, calculate the updated means and change the clusters' centers:

$$m_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{x_j \in C_i^{(t)}} x_j$$

Repeat the above steps until the algorithm comes to convergence (when the means no longer change), such that $C_{t+1} = C_t$.

2.1.2 Complexity

Given the above steps, the k-means algorithm can be shown to have a complexity of $O(n * k * i)$, where n is the number of elements in the set S, k is the number of centroids, and i is the number of iterations. This is a linear complexity that is considered to be very fast. A study has shown that the k-means algorithm requires exponentially many iterations, even in $\mathbb{R}^2$, and has a best known lower bound of $2^{\Omega(n)}$ iterations (Vattani).

2.2 Proof of Convergence

Looking at just the simple method of k-means above, we have enough information to prove that the centroids will always converge in a finite number of iterations. We do this by looking at how each step of the algorithm reduces what we will define as the **cost function**, or how far our centroids are from the geometric mean of the cluster.

Proof. Define the cost function such that $cost(C; z) = \sum_{x \in C} d(x, z)$ and is minimized when $z = mean(C)$, where,

$$mean(C) = \frac{1}{|C|} \sum_{x_j \in C} x_j$$

Let $m_1^{(t)}, ..., m_k^{(t)}, C_1^{(t)}, ..., C_k^{(t)}$ respectively denote the centers and clusters at the start of the tth iteration of k-means. The first step of the iteration assigns each data point to its closest center; therefore

$$cost(C_1^{(t+1)}, ..., C_k^{(t+1)}; z_1^{(t)}, ..., z_k^{(t)}) \leq cost(C_1^{(t)}, ..., C_k^{(t)}; z_1^{(t)}, ..., z_k^{(t)}).$$

After each cluster is re-centered at its mean:

$$cost(C_1^{(t+1)}, ..., C_k^{(t+1)}; z_1^{(t+1)}, ..., z_k^{(t+1)}) \leq cost(C_1^{(t+1)}, ..., C_k^{(t+1)}; z_1^{(t)}, ..., z_k^{(t)})$$

The resulting sequence of cost-value per step is by definition monotonically decreasing. Since this monotonically decreasing sequence is bounded below (by 0), the k-means algorithm converges. □

2.3 The K-Clustering Problem

The K-Clustering Problem is as follows: given a dataset S, find the most optimal k-clustering of the dataset such that the resulting clusterings most accurately represent the actual desired clusterings. This is often measured using an error metric, such as $|1 - \frac{d(A)}{d(C)}| * 100\%$, where A represents the location of the actual measured geometric center of the desired cluster shape, and C represents the center found by the algorithm (measured from the origin). For example, outliers could increase the error in the algorithm because it moves the calculated center farther from the actual center. The k-means clustering problem arises from the fact that the output of k-means is dependent on both our starting k value and the initial location of our centroids. To address this problem, we should first look at some methods of initialization.

2.4 Initialization

A defining feature of k-means is the k parameter itself. The initialization of k, the number of centroids to find in the dataset, must be set before running the algorithm. There are many various methods for accomplishing this. Three common methods are **Dynamic**, **Random**, and **Forgy** method. A more advanced method that is more careful with the picking of the initial k points is **Macqueen Method**, which can be used to give more useful insights into the geometric shaping of the final clusterings.

2.4.1 Dynamic Method

The dynamic initialization method takes k data point from the indexed dataset S and sets the initial centroids equal to those data points. The exact values chosen to be the initial representatives are up to the user. The number of ways to choose k initial clusters from p total data points in the dataset S is defined as $\binom{p}{k}$.

2.4.2 Random Method

The random method iterates through the data points in the dataset S and assigns each of them to a subset of S, C_k, such that there are k subsets with average cardinality $\left\lfloor \frac{|C|}{k} \right\rfloor$. The k initial centroids are then constructed by calculating the mean point value of each C_k and assigning that value as an initial centroid.

2.4.3 Forgy Method

Forgy method is a variation of the dynamic method. Forgy, using the data points from the original dataset S, randomly chooses k data points and assigns each to be the initial centroids.

2.4.4 Macqueen Method

Macqueen's method consists of a random number generator for a given density distribution, a sorting algorithm, and an averaging and update step. This method can be used in combination with k-means to sort the dataset into its *Voroni regions* and to create a more specific clustering associated with the **Centroidal Voroni Tessellation (CVT)** as defined below:

Voroni region: Given a set of data points $\{z_i\}_{i=1}^k$ belonging to dataset C, the Voroni region V_i corresponding to the point z_i consists of all the points in the domain that are closer to z_i than any other point in the set.

Centroidal Voroni Tessellation (CVT): Given the set of points $\{z_i\}_{i=1}^k$ in a dataset C and a density function p, a Voronoi tessellation is called a centroidal Voronoi tessellation (CVT) if

$$z_i = z_i^*, \quad i = 1, ..., k,$$

where z_i^* is the mass centroid of the associated Voronoi region, such that $z_i^* = \frac{\int_V yp(y)dy}{\int_V p(y)dy}$ for any region $V \subset C$.

The Macqueen's k-means method serves to partition the dataset into its CVT by applying the following version of the k-means algorithm:

Macqueen's k-means method: Let C be a closed set, k be a positive integer number of centroids, and p be a probability distribution on C. Generate k random points $\{z_i\}_{i=1}^k$ according to the probability distribution p. Then, initialize indices $j_i = 1$ for all $i = 1, 2, ..., k$. Next, randomly acquire a point $x \in C$ according to the probability distribution p. Then find a z_i that is closest to x and update the z_i element by

$$z_i = \frac{j_i * z_i + x}{j_i + 1}.$$

Repeat the last three steps until a stopping criterion $j_i = j_{i+1}$ is met.

2.4.5 Complexity

Extrapolating from the fact found in Macqueen's K-means method that the dataset can be represented as Voronoi partitions, if the number of centroids k and d (the dimension) are fixed, the k-clustering problem can be exactly solved in time $O(n^{dk+1})$, where n is the number of entities to be clustered (Inaba). While anything of power $O(n^2)$ and higher is considered to be suboptimal inefficiency, this method does exactly calculate an optimal solution to the k-clustering problem.

2.4.6 Inaba's Proof of K-Means-Voronoi Equivalence

Theorem: *The number of Voronoi partitions of n points by the Euclidean Voronoi diagram generated by k points in the d-dimensional space is $O(n^{dk})$, and all the Voronoi partitions can be enumerated in $O(n^{dk+1})$ time.*

Proof. Consider the (dk)-dimensional vector space consisting of u with $u = (u_1, u_2, ..., u_k)$. In this (dk)-dimensional space, we can define an equivalence relation among points such that two points are in the equivalence relation if their corresponding Voronoi partitions are identical. The equivalence relation produces a subdivision of this space into equivalence classes. For each pair of distinct u_{j_1} and u_{j_2}, among u_j $(j = 1, ..., k)$ and each point $p_i = (x_i)$ among p_i $(i = 1, ..., n)$, consider an algebraic surface in this (dk)-dimensional space defined by

$$||x_i - u_{j_1}||^2 - ||x_i - u_{j_2}||^2 = 0$$

where x_i is regarded as a constant vector. The number of such surfaces is $nk(k-1)/2$. The arrangement of these $nk(k-1)/2$ algebraic surfaces coincides with the subdivision defined by the equivalence relation from Voronoi partitions. The number of Voronoi partitions is bounded by the combinatorial complexity of the arrangement of $nk(k-1)/2$ constant-degree algebraic surfaces, and the theorem follows. $\square$

2.5 Discussion

2.5.1 Limitations

K-means, while customizable, inherently has a plethora of limitations. For example, it can only work with euclidean distances, thus limiting the kinds of data sets that the algorithm can work with. Also, the number of clusterings, k, must be chosen beforehand and cannot be updated during the course of the algorithm. Choosing too few k points could also result in inaccurate clusterings. The algorithm also treats each data point equally with regards to their weight. If we wanted certain data points to have more pull on its centroid's location, there is no included method to do this; one must be constructed (i.e.: Macqueen's algorithm). Inversely, outlying points can move the centroid away from its "true" location because the algorithm gives equal weight to outlying points as more dense, truly-clustered points.

2.5.2 Improvements

With the above stated, there are multiple methods that have been developed for improving and customizing the k-means algorithm. The previously mentioned Macqueen's algorithm seeks to add weights to points using a predefined probability distribution. The k-means++ algorithm seeks to add a dynamic step to either add or subtract centroids at each step of the algorithm. To better reduce the effects of outliers other algorithms on certain datasets, other clustering algorithms, like DBSCAN (discussed later), could be run on the dataset to identify and eliminate outliers before running k-means.

3 Connectivity Based Clustering

Connectivity Based Clustering, also known as **Hierarchical Clustering**, has to do with grouping data based on their distance from each other. In other words, instead of partitioning the data, clustering with this method creates various levels and/or ranks of data sorted by distance from each other. This kind of sorting algorithm is particularly useful on data sets that have some sort of connection to the other points and can be used to identify and analyze "hubs" in datasets. Real-life examples of a good dataset for connectivity based clustering might be the number of clicks (paths) to get from Facebook to Wikipedia, or all domestic flights in the United States. There are two general types of connectivity based clustering: **Agglomerative** and **Divisive**.

3.1 Agglomerative Hierarchical Clustering

This form of clustering considers every data point its own cluster, then merges these clusters based on a given metric criteria. For this reason, it is often considered a bottom-up algorithm, looking at the data at individual points and then combining, or agglomerating, the data points into clusters (where it gets its name) based on the metric-linkage criterion.

3.2 Divisive Hierarchical Clustering

In some ways, this method is the opposite of agglomerative. Divisive hierarchical clustering looks at the the dataset as a whole, defining it as one cluster. Then, based on the metric-linkage criterion, the algorithm splits the whole set into subsets with hierarchy, continuing recursively until completion.

3.3 Metric-Linkage Criterion

The Metric-Linkage Criterion has to do with linking data points and clusters in the dataset based on the distance between them in order to create larger or smaller clusters.

3.3.1 Metric Definition

We will be using our predefined metric function $d(x, y)$ when talking about $x, y \in S$ where S is our dataset. The distance between two data points in this context will be defined as their **dissimilarities**. The distance is often expressed using euclidean distance, but can also be represented by other, non-geometric measurements (i.e.: number of clicks to get from one web page to another).

3.3.2 Establishing Links

Let S be a dataset composed of data points $\{x_1, ..., x_n\}$. Let the dissimilarity between each point x_i and x_j in S be the distance between the points, $d_{ij} = d(x_i, x_j)$.

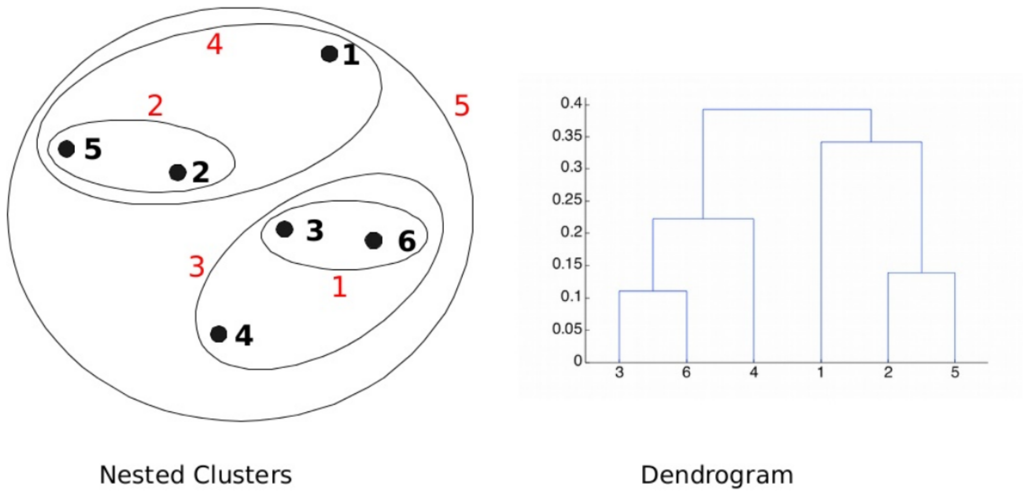
Furthermore, let S be partitioned into g subgroups. Two subgroups S_{g_i} and S_{g_j} are said to have a **dissimilarity score** of $min(d(S_{g_i}, S_{g_j}))$, which is the minimum distance between the points of the two subgroups. If this dissimilarity score is equal to or less than a defined similarity score, the groups are said to be **linked**.

3.4 Agglomerative Example (The CLINK Algorithm)

3.4.1 CLINK Algorithm:

Complete **Link**age Clustering:

Initialization: Partition S into G_k subgroups, where $k = |S|$, such that each element of S is in its own subgroup G_k. Then for all pairs of subgroups G_i^1 and G_j^1, where $i, j \leq k, \in \mathbb{Z}^+$ and $i \neq j$, merge them together to form $G_{(i,j)}^2 = G_i^1 \cup G_j^1$, such that they have the minimal dissimilarity score between them. i.e: $min(d(G_i^1, G_j^1))$. Repeat the merges for G_i^2 and G_j^2 and $k - 1$ until only one group remains.



An example of CLINK performed on the dataset on the left (Euclidean distance metric)

3.5 Discussion

3.5.1 Distance Metrics

To cluster the data, we must first identify what **distance metric** we will use on the data set to analyze distances between the points and to establish links (explained below). Because connectivity based clustering gives heavier weight to data points that are more closely connected to each other, the kinds of metric used to measure distances plays a large role in the formation of clusters than other algorithms. For example, Manhattan distance and Euclidean distance between the points $(0, 0)$ and $(0, 1)$ is 2 and$\sqrt{2}$ respectively, and yet they can yield very distances between data points and therefore equally different clusterings of the dataset. The distance could also be defined as something more arbitrary, like the number of edges between one vertex and another, where the algorithm does not care about the actual geometric length of the edges and is therefore not restricted to geometric distance.

3.5.2 Complexity

The complexity of complete-link clustering has an upper bound of $O(n^2 \log n)$. Some members of the mathematical community have constructed a complete-link clustering method with complexity $O(n^2 \log n)$. This is done by "computing the n^2 distance metric and then sort the distances for each data point, giving $O(n^2 \log n)$. After each merge iteration, the distance metric can be updated in $O(n)$. We pick the next pair to merge by finding the smallest distance that is still eligible for merging. If we do this by traversing the n sorted lists of distances, then, by the end of clustering, we will have done n^2 traversal steps. Adding all this up gives you $O(n^2 \log n)$"

(Abdullah).

3.5.3 Limitations

The complexity of the most common hierarchy based algorithms is generally exponential (around $O(n^2)$ and $O(n^3)$). This makes these kinds of algorithms suboptimal to use on large datasets, and customization is needed to adapt the algorithm to the dataset, or work needs to be done on the dataset beforehand to make it more friendly for agglomerative hierarchal sort.

4 Density-based clustering

Density Based Clustering is a sorting method used to identify and cluster areas in a dataset that contain dense amounts of data points, separated by areas with more spacial data point groupings. The density of a cluster is determined using a defined distance metric, with the clusters themselves defined as a maximal set of density-connected points. This method works well on datasets that are expected to not contain much variance in their densities, and are used to filter out noise or identify concentrations of data points. Some real life datasets that density based clustering has proved useful on include filtering for cancer cells and identifying black holes.

4.1 DBSCAN

DBSCAN is an extremely popular density based sorting algorithm used to identify dense areas of data points using what will be defined as having many neighbors in their e-**neighborhoods**, as well as identifying low-density outliers who have fewer neighbors.

4.1.1 e-Neighborhoods

The e-neighborhood N_e around a data point p in the dataset S can be defined as the number of data points within a radius e around p, such that

$$N_e(p) : \{q|d(p,q) \leq e\}$$

Where $q \in S$ and d is the distance between p and q in a defined metric space of S.

- $N_e(p)$ is said to have **high density** if it contains at least an m points, where m is a real number.
- $N_e(p)$ is said to have **low density** if it contains less than an m points, where m is a real number.

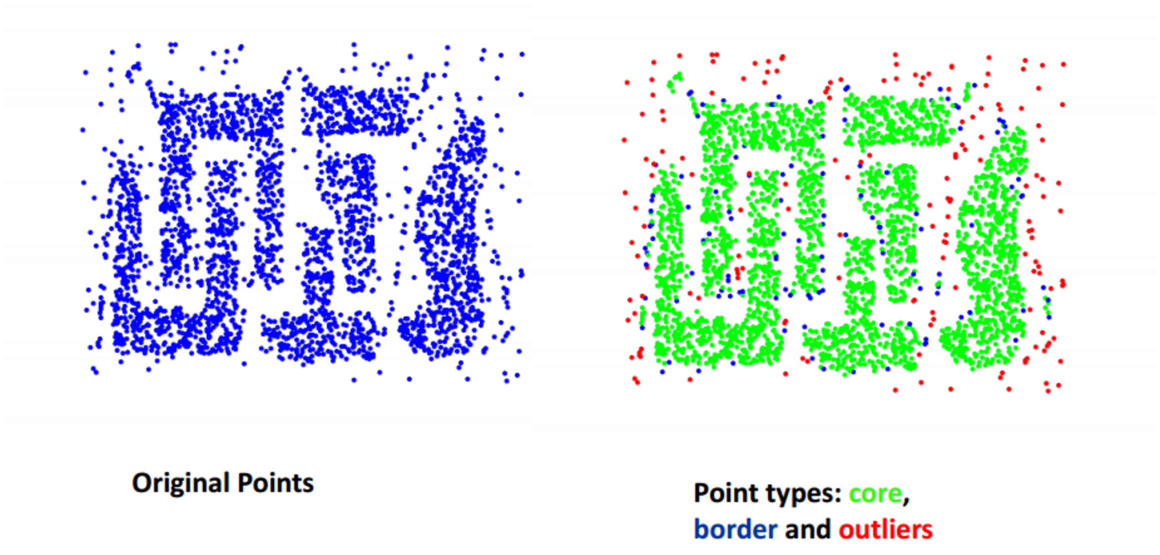
4.1.2 DBSCAN Algorithm Preliminaries

Given the above information, we are able to use DBSCAN to classify each data point into one of three different classifications, which can be represented as colorings:

Let e and m be defined, then

- **Core Point:** if a data point p has an e-neighborhood with high density, it is classified as a core point.
- **Border Point:** if a data point p has an e-neighborhood with low density but resides in the e-neighborhood of a core point, it is classified as a border point.

- **Noise Point:** if a data point p has an e-neighborhood that has low density and resides outside the e-neighborhood of a core point, it is classified as a noise point.



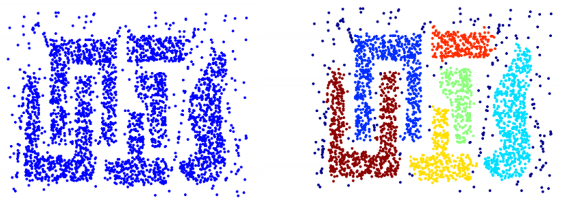
An example of DBSCAN run on the left dataset, with $e = 10mm$ and $m = 4$.

4.2 Discussion

4.2.1 Advantages

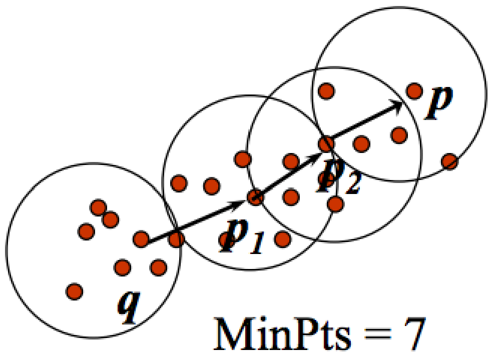
Unlike k-means, DBSCAN can find a dynamic number of clusters, given that the densities in the clusters stay relatively unvaried. These clusterings can be arbitrary shapes, as shown above, and can even be connected to other clusters. On a dataset with relatively unvaried density, we could run DBSCAN, eliminate outliers, then run k-means on the dataset.

4.2.2 Improvements



Original dataset after partitioning with respect to individual clusterings during the assignment step.

A problem that arises with the simple form of DBSCAN above is that the maximal set of core points acquired are all isomorphic to each other. There is no way to tell which clustering is which, only what points belong to which cluster. Given the parameters for which the algorithm was run, specifically e and m, we can run an algorithm like **Prim's Algorithm** to find the connected core points, starting at any core point, and partition them into subsets. This is possible because the relationship between core points and their e-neighborhoods is equivalent to the relationship between the construction of edges in a connected graph given the e-m criterion. In DBSCAN, this is called being **density reachable**, and is best performed during the assignment step.



p is (indirectly) density-reachable from q.

5 Conclusion

In conclusion, there are many different types of clustering algorithms who have varying uses. The success of the algorithm on clustering the data largely depends on the dataset itself (do the elements have euclidean properties?) as well as the geometric shapes and densities of the clusters within the dataset. K-means tends to work well on datasets with more obvious, spherical clusterings of varying density with few outliers. Hierarchical clustering like CLINK works well on datasets whose elements might not adhere to the notions of euclidean geometry. DBSCAN does well with datasets containing more ambiguous clusterings and takes outliers into account, as long as the densities between the various clusters are fairly similar. All data clustering algorithms have their strengths and weaknesses, and many methods have been developed to improve their outcomes, like Macqueen's k-means algorithm and DBSCAN with density-reachable clusterings. The field of data analysis is a relatively new mathematical field, and it is very likely that these very popular algorithms will continue to be built upon in their future.

Works Cited

- Abdullah, Ahsan, and Ahmad Barnawi. “A Novel Crossing Minimization Ranking Method.” Applied Artificial Intelligence 29.1 (2015): 66-99. Web.
- Celebi, M. Emre, Hassan A. Kingravi, and Patricio A. Vela. “A Comparative Study of Efficient Initialization Methods for the K-means Clustering Algorithm.” Expert Systems with Applications 40.1 (2013): 200-10. Web.
- Defays, D. “An Efficient Algorithm for a Complete Link Method.” The Computer Journal 20.4 (1977): 364-66. Web.
- Du, Qiang, and Tak-Win Wong. “Numerical Studies of MacQueen’s K-means Algorithm for Computing the Centroidal Voronoi Tessellations.” Computers & Mathematics with Applications 44.3-4 (2002): 511-23. Web.
- Gan, Guojun. Data Clustering in C an Object-oriented Approach. Boca Raton: Chapman and Hall / CRC, 2011. Print.
- Inaba, Mary, Naoki Katoh, and Hiroshi Imai. “Applications of Weighted Voronoi Diagrams and Randomization to Variance-based K-clustering.” Proceedings of the Tenth Annual Symposium on Computational Geometry - SCG ’94 (1994): n. pag. Web.
- Kaufman, Leonard, and Peter J. Rousseeuw. Finding Groups in Data: An Introduction to Cluster Analysis. Hoboken: Wiley Interscience, 2005. Print.
- Mumtaz, Et Al, K. “An Analysis on Density Based Clustering of Multi Dimensional Spatial Data.” Indian Journal of Computer Science and Engineer-

ing. Vivekanandha Institute of Information and Management Studies, Aug. 2012. Web.

- Reddy, Damodar, and Prasanta K. Jana. “Initialization for K-means Clustering Using Voronoi Diagram.” Procedia Technology 4 (2012): 395-400. Web.
- Vattani, Andrea. “K-means Requires Exponentially Many Iterations Even in the Plane.” Proceedings of the 25th Annual Symposium on Computational Geometry - SCG ’09 (2009): n. pag. Web.