Modeling Clusters For Data Mining Prediction

Clint Cooper CLINTGCOOPER@GMAIL.COM
Department of Computer Science
Montana State University
Bozeman, MT 59715, USA

Emily Rohrbough EMILYROHRBOUGH@YAHOO.COM
Department of Computer Science
Montana State University
Bozeman, MT 59715, USA

Leah Thompson LEAH.THOMPSON@MSU.MONTANA.EDU
Department of Computer Science
Montana State University
Bozeman, MT 59715, USA

Abstract

This paper evaluates five unique methods for clustering data: k-means, DB-Scan (density-based spatial clustering of applications with noise), competitive learning neural network, particle swarm optimization (PSO), and ant colony optimization (ACO). All five clustering methods were applied to ten data sets and performance was evaluated using cluster cohesion and separation. It was predicted that k-means would create highly cohesive clusters high dimensional data sets, DB-Scan would create less clusters and converge faster for high dimensional data, competitive learning neural network would have high cohesion for high dimensional data sets with a low number of clusters, ACO would produce cohesive clusters but have poor separation, and PSO would produce clusters that are separated well but converge slowly for lower dimensions. A total of 50 tests were performed for the five clustering techniques over each of the ten data sets. It was concluded more tests were needed with varying tunable parameters to make valid conclusions for each of the algorithms over the data sets.
Keywords: Data Mining, Clustering Algorithms, K-Means, DB-Scan, Competitive Learning Neural Network, Particle Swarm Optimization, Ant-Colony Optimization

1. Introduction

Data mining is the analytical process of exploring data to search for consistent patterns and systematic relations between variables (Dell, 2015). This process is used to develop a model to make future predictions regarding similar data (Dell, 2015). Data mining consists of three stages: exploration, model building, and deployment (Dell, 2015), and is useful for determining associations, sequences, classifications, clustering and forecasting (Rouse, 2008). Typically, the emphasis of data mining in regards to machine learning is the accuracy of the prediction made by the model, not whether the model is interpretable (Dell, 2015). Thus, for this project, the performance of five algorithms was evaluated when given the data mining task of clustering data based on partitions and density (Sheppard, 2015b). The five algorithms used for this project were k-means, DB-Scan, competitive learning neural network, ant-colony optimization and particle swarm optimization. K-means and DB-Scan are algorithms that were specifically designed for clustering data. The competitive learning neural network, particle swarm optimization and ant-colony optimization algorithms are also capable of clustering data, however, to cluster data, they require a clustering method to be incorporated into their implementation.

2. Problem Statement and Hypothesis

Clustering is the main multi-objective task of data mining, such that data is placed in groups based on their similarities according to some clustering criteria (Tan et al., 2006). The notion of a cluster cannot be precisely defined, but is a direct result of the clustering algorithm implemented (Tan et al., 2006). This project only focused on partition-based and density-based clustering tasks for the k-means, DB-Scan, competitive learning neural network, particle swarm optimization, and ant-colony optimization algorithms. Partition-based clustering is simply a division of a set of data into non-overlapping clusters such that each data object belongs in exactly one cluster(Tan et al., 2006). Density-based clustering is the process of separating similar data into dense, concentrated regions that are surrounded by sparse regions of noise (Tan et al., 2006). Running each algorithm results in a clustering model that can be used for future clustering predictions. For each of the ten data sets obtained from the UCI repository, the convergence rate and prediction accuracy of the produced clustering model were used to evaluate each algorithm.

Given the clustering methods of the implemented algorithms, it is not possible to directly compare the clustering tendencies of each algorithm because each method clusters according to the type of data it was designed for. However, since almost every clustering algorithm will find clusters in a data set even if that data set has no natural clustering structure (Tan et al., 2006), we can hypothesis for each algorithm regarding cohesion and separation factors.

- **K-Means:** Starting with the clustering performance, the algorithm is expected to create highly cohesive clusters for data sets with high dimensionality. It is thought more attributes for each element in the data set will help the algorithm select the distribution that best fits that data point. In terms of convergence, lower dimensionality is expected to increase the convergence of k-means (Mishra).

- **DB-Scan:** The DB-Scan algorithm is expected to create less clusters and converge faster for data sets of lower dimensionality (Mishra). It is thought more attributes will increase the reachability and connectivity of the data set, a direct influence on the number of clusters formed with DB-Scan. However, the convergence with lower dimensions is also thought to be fast given the calculation of reachability and connectivity will not need to be repeated for several iterations.

- **Competitive Learning:** It is expected that the competitive learning algorithm will perform and converge in an almost identical manner to the k-means algorithm given the similarities between the two techniques. However, the competitive learning neural network will likely form less clusters since the specified maximum number of cluster does not have to be reached.

- **Ant Colony Optimization:** The ant colony optimization algorithm is expected to produce clusters of high cohesion with poor separation because the ant behavior focuses on creating tight groups of similar data points and does not move dissimilar data points away from each other. Additionally, the convergence of the algorithm is expected to be directly effected by the number of points in the data set so more data points would result in slower convergence.

- **Particle Swarm Optimization:** It is predicted that the clusters formed by the particle swarm optimization algorithm will be well separated given the fitness function. In addition, since the maximum number of specified clusters does not have to be reached, each cluster should be fairly cohesive. The convergence of the algorithm is expected to be slower for higher dimensions because the search space of the particles expands proportionate to the dimensionality.

3. Algorithms Implemented

3.1 K-Means

The k-means algorithm is a centroid-based partitioning technique for clustering that attempts group n data points into k user-specified clusters (Tan et al., 2006). This approach tends to follow a Gaussian model and has two main steps after initializing the centroids: (1) the assignment step and (2) the update step (MacKay, 2003). The initial centroids are k-random samples of the input points.

Next, the assignment step places each data point with the centroid that has the closest mean. This is calculated as follows (Sheppard, 2015a):

$$argmin_c \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2$$

where C_i is the set of points in cluster i, μ_i is the centroid of cluster i, and x is the input vector (Sheppard, 2015a). This step forms the clusters.

The update step then re-calculates the centroids of each cluster to match the sample means of the data points that have been assigned to the given cluster (MacKay, 2003). The centroids are updated as follows (Hugo, n.d.):

$$\mu_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j$$

The k-means algorithm repeats these two step steps until one of the follow stopping conditions are met: (1) no element moves clusters, (2) no change in the centroid values, or (3) the maximum number of iterations is reached (Baswade and Nalwade, 2013).

3.2 DB-Scan

The Density Based Spatial Clustering of Applications with Noise algorithm (M. Ester and Xu), or DB-Scan, is a density-based, partitioning algorithm for clustering that relies on the relative densities of the data points to determine clusters, rather than assuming the data follows a Gaussian model (Sheppard, 2015b). In this context, density represents the number of points within a user-specified radius, ε (M. Ester and Xu). DB-Scan separates data points into three types, core points, border points, and noise points (Sheppard, 2015b). Core points contain at least the user-specified number of minimum points, $MinPts$, within ε, while border points are defined as data points that fall within the ε-neighborhood of a core point (Sheppard, 2015b). This is referred to as a high-density region (Tan et al., 2006). Noise points are anything beyond ε in low-density regions (Tan et al., 2006) and are discarded (Sheppard, 2015b), making DB-Scan an incomplete clustering of the data points (Tan et al., 2006).

The DB-Scan algorithm takes an arbitrary, unvisited data point and forms a cluster if the point satisfies the requirements of a core point(M. Ester and Xu). When a cluster is formed, every other point found within the ε-neighborhood is also included in the cluster and is marked as visited. This process continuously cascades into each point's neighborhood until the entire cluster is found (M. Ester and Xu). This cascading behavior is the application of density reachability and density connectivity (M. Ester and Xu), which directly effects the number of clusters produced by the algorithm (Tan et al., 2006).

3.3 Competitive Learning Neural Network

A competitive learning neural network is comprised of two, fully connected layers, an input layer and a competitive output layer (Prism). This algorithm is similar to the k-means clustering algorithm (Sheppard, 2015a). The neural network is given normalized input vectors of n-dimensional space and unsupervised learning is applied to determine which output node best represents the vector (Prism). This learning aspect of the neural network, called competitive learning, is where the k output nodes represents

a cluster and a winner-takes-all approach is used to update the weight vector, $w_{j,i}$, associated with the input node and the best cluster node, j (Prism)(Sheppard, 2015a).

For the winning node, the scalar product $x_i \dot{w}_{j,i}$ of the normalized input vector, x_i, and the weight vector is equal to the cosine of the angle spanned by both vectors Prism. This means the weight vectors for the network always sum to one and the winning weight vector is updated to rotate towards x_i. The winning weight vector is updated accordingly:

$$w_{i,j} = w_{i,j} + \eta(w_{i,j} - x_i)$$

where η is a user specified learning rate between 0 and 1. As more input vectors are analyzed, the weights rotate to reflect the clusters formed (Prism). The update weight function for the winning node utilizes the distance between the weight and input vectors (Sheppard, 2015a). Given the nature of the winner-takes-all approach, it is possible that less clusters are formed then the k cluster nodes initially specified in the competitive layer (Sheppard, 2015a).

3.4 Ant Colony Optimization Clustering

Ant Colony Optimization (ACO) is an algorithm was inspired by the behavior of ants; it employs agents, or ants, that move around a grid search space searching for data items that are isolated or dissimilar to the data items around it (Sheppard, 2015a). When an ant locates a data point that matches either of these criteria, the ant will it pick up and move it through the space until similar items are found, where the ant then drop it(Sheppard, 2015a). This behavior eventually creates dense clusters with similar characteristics. When ants base these pickup and drop operations on the environment, rather than on any pheromones laid down by other ants, the process is known as ant-based clustering (Handl et al., 2003).

The ACO implementation for this project randomly placed the data points and ants in a nxn search space that was derived from the square of the number of inputs time ten. Then, during each iteration, each ant had the opportunity to move itself, pickup a data point or drop a data point. The probability that an ant would pick up a data point, i, was calculated as follows (Handl et al., 2003):

$$ProbPick(i) = \begin{cases} 1 & if Fitness(i) \leq 1 \\ \frac{1}{Fitness(i)^2} & otherwise \end{cases}$$

Where the probability that an ant would drop a data point is (Handl et al., 2003):

$$ProbDrop(i) = \begin{cases} 1 & if Fitness(i) \geq 1 \\ Fitness(i)^4 & otherwise \end{cases}$$

If an ant neither picked up or dropped a data point it would move. The fitness function used to determine the fitness of a data point, given its location, also influenced what action an ant should take. This was calculated as follows (Handl et al., 2003):

$$Fitness(i) = \begin{cases} \sum_j (1 - \frac{d(i,j)}{activity}) & if(Fitness(i) > 0 \wedge \forall j(1 - \frac{d(i,j)}{activity}) > 0) \\ 0 & otherwise \end{cases}$$

where j are the other data points within i's neighborhood, d(i,j) is the Euclidean distance between i and its neighbors, and *activity* is the measure of how successful that particular ant is at placing data. The neighborhoods surrounding an ant was dynamically defined as the number of squares surrounding a location, where the range was increase from 1 square for the first 20 percent of the iterations and expanded by 1 square for each additional 20 percent completed.

The success rate of the ants were initially randomized and were updated as the ants moved data points around the search space. The success of each ant was calculated every ten data drop and was updated as

follows(Handl et al., 2003):

$$activity = \begin{cases} activity + 0.01 & if \frac{N_{bads}}{N_{total}} > 0.75 \\ activity - 0.01 & otherwise \end{cases}$$

where N_{bads} is the number of times a data point was dropped in a less optimal location and N_{total} is the total number of drops performed.

3.5 Particle Swarm Optimization Clustering

Particle Swarm Optimization is a population based stochastic optimization algorithm that was inspired by the behavior of flocking birds (Chuang et al., 2012). In this algorithm, particles, which represents the potential solutions, explore the search space by following the path of the current optimum. Each particle tracks its best fit solution, called *pbest*, and also references the best solution found so far, called *gbest*. The position, x, and velocity, v, of the particles are updated in correlation to *pbest* and *gbest* for each iteration, such that its velocity represents how much that particle needs to move in order to reach that target solution. The particles continue to move through the search space until a specific criteria is met or the maximum number of iterations is met.

The fitness function used to evaluate a particle's position is the summation of Euclidean distances for each data point to the nearest proposed cluster.

$$Fitness(x) = \sum_{i=1}^{|inputs|} \sum_{j=1}^{|inputs|} d(p_i, c_j)$$

where p_i is every element in the input set and c_j is the corresponding closest cluster in the *gbest* particle. The PSO algorithm implemented for this project also utilized a Gauss chaotic map to avoid being stuck in a local optimum. Gauss chaotic maps allows a complete analysis of qualitative and quantitative properties of chaos such that dense periodic points, mixing and sensitivity properties can be applied for adaptive action. Thus, the use of the Gauss chaotic map affects how the velocity of a particle is updated:

$$v_i' = wv_i + Gr_1(pbest_i - x_i) + Gr_2(gbest - x_i)$$

where w is the inertia factor that prevents velocity from exceeding the search space, and Gr_1 and Gr_2 are values between 0 and 1 found from the Gauss chaotic map. Gr_1 and Gr_2 were randomly initialized, but updated the same using the following recursive equation:

$$Gr(x) = \begin{cases} 0 & Gr(x) = 0 \\ \frac{1}{x} \bmod 1 & Gr(x) \in (0, 1) \end{cases}$$

where x is the previous Gr result. This algorithm replaces the need for phi_1 and phi_2 used in the traditional PSO update equation. Once a new velocity has been found, the position of a particle is updated as:

$$x_i' = x_i + v_i$$

4. Experimental Approach

Ten data sets were obtained from the UCI Machine Learning Repository and used for the clustering analyses. The data sets were chosen based on the following conditions: (1) must have no missing data, (2) must have 4-10 attributes, and (3) must have more than 400 instances. These constraints were placed on the choice of data sets in hopes to compile test results with similar clustering output within consistent convergence rates due to the similar data set characteristics. The characteristics of each data set can be viewed in Table 1. Note: each data set set was normalized to real valued representation.

Data Set	Num Instances	Num Attributes
Balance Scale (Siegler, 2014)	625	4
Blood Transfusion Center (Yeh, 2008)	748	4
Car Evaluation (Bohanec, 1997)	1,728	6
Contraceptive Method Choice (Lim, 1997)	1,473	9
Indian Liver Patient (Ramana, 2012)	583	10
Tic-Tac-Toe Endgame (Aha, 1996)	958	9
User Knowledge Modeling (Kahraman et al., 2009)	403	5
Wilt (Johnson, 2013)	4,889	6
Wholesales Customers (Cardoso, 2014)	440	8
Yeast (Nakai, 1996)	1,484	8

Table 1: The details of the ten data sets chosen from the UCI repository (Bache and Lichman, 2015).

There was a total of 50 tests performed. 10 tests were run for each of the five algorithms over each of the 10 selected data sets. For each test, internal validation techniques were used for evaluating the performance of each algorithm. Internal evaluation is the unsupervised method for evaluating the validity of the clustering results without referencing any external information (Tan et al., 2006). The cluster characteristics considered for this internal evaluation are cluster cohesion and cluster separation. Cluster cohesion is a measure of how similar, or far away, the points in a cluster are to each other Sheppard (2015a). The cohesion of a cluster is calculated as follows (Sheppard, 2015a):

$$coh(C_i) = \sum_{x \in C_i, y \in C_i, x \neq y} \frac{d(x,y)}{|C_i|}$$

where $d(x,y)$ is the euclidean distance between points x and y within cluster C_i. Ideally the cohesion value of a cluster will be small, indicating the cluster contains more similar data. The cluster separation characteristic is the measurement of how well an algorithm was able to distinctly split data into different clusters. The separation between two clusters is calculated as follows (Sheppard, 2015a):

$$sep(C_i, C_j) = \sum_{x \in C_i, y \in C_i} d(x,y)$$

where $d(x,y)$ is the euclidean distance between the point x in cluster C_i and the point y in cluster C_j. If the clusters are well separated clusters, large separation values will be returned, while overlapping clusters will return considerably small separation values.

4.1 Tuning Process

The following sections specify the parameters used for each training algorithm implemented.

- **Competitive Learning Neural Network:** There are three tunable parameters in the competitive learning network: (1) the number of output nodes, (2) the learning rate, and (3) the number of iterations to reach termination. For this implementation, we chose 20 nodes for the output layer with a 0.05 learning rate and 100,000 iterations. These parameters resulted in provide a cluster prediction for the other algorithms that required cluster specification that as an input.

- **K-Means:** There two tunable parameters used for this k-means algorithm implementation: (1) the specification of the number of clusters, k, and (2) the number of iterations to reach termination. For this implementation, the number of clusters was set to one greater than the number of clusters generated by Competitive Learning and the number of iterations was set to 10,000.

- **DB-Scan:** There are two tunable parameters used for this DB-Scan implementation: (1) the neighborhood threshold, ε and (2) the minimum number of points needed to from a cluster, $MinPts$ (Ester et al., 1996). For this implementation, ε was set to 20% of the distance of the two furthest input points. $MinPts$ was set to 15.

- **Ant Colony Optimization:** There are two tunable parameters for this implementation of the Ant Colony Optimization algorithm: (1) the number of ants and (2) the number of iterations for the system. Ants was set to 10 and iterations was set to 1000. Number of ants was set to 10 to reduce excessive influence on the overall fitness calculations caused by too many ants.

- **Particle Swarm Optimization:** There are three tunable parameters for this implementation of the Particle Swarm Optimization: (1) the number of predicted clusters (2) the number of particles and (3) the number of iterations. Number of predicted clusters was set to one greater than the number of clusters generated by Competitive Learning. The number of particles was set to three times the number of inputs and the number of iterations was set to 100.

5. Results

The results are explored below. For each algorithm and each data set the following was calculated: the number of clusters formed, the average number of particles per cluster, the cohesion value of the clusters and the separation value of the clusters. The average number of particles per cluster had no significant contribution to our results, however, it was interesting to see. Additionally, each algorithm reached the maximum number of iterations for each data set and the maximum number of iterations per algorithm was inconsistent. Given this, no meaningful interpretation of the convergence rates of the algorithms can be made. Either consistent interactions or multiple runs with varying parameters would have been needed for good convergence analysis.

5.1 K-means Results

K-Means Results				
Data Set	**No. Clusters**	**Mean±Stdev**	**Cohesion**	**Separation**
Balance Scale	7	89.29±8.99	388.4	168608.87
Blood Transfusion Center	14	53.43±33.51	8.14	29325.02
Car Evaluation	6	288±37.33	1399.79	1428629.36
Contraceptive Method Choice	6	245.5±104.1	125.51	10548.35
Indian Liver Patient	10	57.9±66.91	12.37	10548.92
Tic-Tac-Toe Endgame	4	239.5±27.62	1369.44	580136.92
User Knowledge Modeling	5	80.6±28.51	214.73	49899.19
Wholesales Customers	20	22±18.15	28.1	17536.38
Wilt	4	1084.75±435.46	426.25	1678071.21
Yeast	5	296.8±175	390.33	337728.42

Table 2: The results of the k-means on the data sets.

The K-means algorithm obtained decent cohesion for most of the data sets. The data sets that obtained the best cohesion had less separation, while some data sets that were less cohesive were better separated. This indicated the tuning process used to determine the number of clusters to form may not have been the best.

DB-Scan Results				
Data Set	**No. Clusters**	**Mean±Stdev**	**Cohesion**	**Separation**
Balance Scale	400	1.56±1.06	79.44	186039.37
Blood Transfusion Center	8	93.50±169.22	38.27	22470.53
Car Evaluation	1296	1.33±0.67	203.41	1633334.65
Contraceptive Method Choice	13	113.31±186.03	143.82	206460.03
Indian Liver Patient	6	96.17±226.75	60.95	7069.14
Tic-Tac-Toe Endgame	0	0±0	0	0
User Knowledge Modeling	118	3.37±4.04	91.04	56995.53
Wholesales Customers	13	33.23±110.54	82.86	6594.85
Wilt	-	-	-	-
Yeast	65	22.28±80.37	321.89	319049.38

Table 3: The results of the DB-Scan algorithm on the data sets. "-" indicates that test did not finish.

5.2 DB-Scan Results

The DB-Scan algorithm produced low cohesion clusters for most data sets, however, the number of formed clusters varied significantly per set. DB-Scan is effective at finding dense regions, which is why low cohesion was expected, but the tunable parameters were sensitive and we could have used a better method for determining the neighborhood epsilon. Since the number of clusters was high and larger than the other algorithms for most data sets, it is thought the tunable parameters need improvement. In regards to separation, most clusters were well-separated. The Indian Liver Patient and Wholesale Customers did not produce overly separated clusters.

5.3 Competitive Learning Results

Competitive Learning Results				
Data Set	**No. Clusters**	**Mean±Stdev**	**Cohesion**	**Separation**
Balance Scale	6	101.17±92.16	513.8	141101.88
Blood Transfusion Center	13	57.54±91.12	13.28	28347.45
Car Evaluation	5	345.6±281.38	1718.78	1173974.48
Contraceptive Method Choice	5	294.6±387.1	194.88	162662.87
Indian Liver Patient	9	64.33±90.71	22.09	10113.02
Tic-Tac-Toe Endgame	3	319.33±382.21	1516.58	265248.73
User Knowledge Modeling	4	100.75±82.55	257.46	39563.78
Wholesales Customers	19	23.16±12.69	55.49	17213.90
Wilt	3	1446.33±2103.99	748.16	55155.91
Yeast	4	371±701.74	532.85	55155.91

Table 4: The results of the competitive learning neural network on the data sets.

The competitive learning neural network appeared to have created low cohesion for almost all data sets, specifically the Car Evaluation and Tic-Tac-Toe sets. However, the best cohesive clusters were obtained with the Blood Transfusion and Indian Liver Patient data sets. The separation of the competitive learning network was fairly high for most data sets. The Car Evaluation data set had some of the most separated clusters although they were not cohesive. The smallest separation is seen in the Indian liver data set which might be related to this data set not having distinct clusters that are close together.

Ant Colony Optimization Results				
Data Set	**No. Clusters**	**Mean±Stdev**	**Cohesion**	**Separation**
Balance Scale	265	2.36±1.87	335.94	185348.20
Blood Transfusion Center	164	4.56±1.94	59.92	29393.41
Car Evaluation	300	5.76±3.21	1523.69	1627534.70
Contraceptive Method Choice	266	5.54±2.89	264.19	234962.68
Indian Liver Patient	126	4.60±2.48	30.09	10954.49
Tic-Tac-Toe Endgame	596	1.61±1.16	588.25	744929.18
User Knowledge Modeling	156	2.58±2.50	178.31	58898.16
Wholesales Customers	79	5.57±2.91	67.55	17708.31
Wilt	-	-	-	-
Yeast	272	5.46±2.69	447.14	407681.02

Table 5: The results of the ant colony optimization on the data sets. "-" indicates that test did not finish.

5.4 Ant Colony Optimization Results

For the ant colony optimization algorithm, the cohesion values do not represent good clustering. This is due to the larger number of clusters containing fewer points on average. This is most likely related to the tunable parameters of ACO being sub-optimal. One of the most surprising clustering results was Tic-Tac-Toe, which contained relatively similar data. It resulted in the largest number of clusters, although well separated. The separation values were high, suggesting that while there were many clusters, they were separate but not always cohesive. The smallest separation was the Wholesale Customers data set.

5.5 Particle Swarm Optimization Results

Particle Swarm Optimization Results				
Data Set	**No. Clusters**	**Mean±Stdev**	**Cohesion**	**Separation**
Balance Scale	7	89.29±10.98	388.44	168534.6
Blood Transfusion Center	2	374.00±521.84	72.02	2938.63
Car Evaluation	6	288.00±35.36	1401.45	1428713.95
Contraceptive Method Choice	1	1473±0	320.36	0
Indian Liver Patient	1	579±0	38.17	0
Tic-Tac-Toe Endgame	4	239.50±27.33	1366.29	580636.3
User Knowledge Modeling	5	80.60±26.59	214.78	50024.32
Wholesales Customers	3	146.67±251.44	77.98	1014.26
Wilt	-	-	-	-
Yeast	4	371.00±613.24	473.7	141827.77

Table 6: The results of the particle swarm optimization on the data sets. "-" indicates that test did not finish.

The particle swarm optimization algorithm generally created fewer clusters than specified as the maximum. The clusters however, dramatically differed in the number of sizes. In some cases, PSO returned a single cluster for the entire data set, other times the algorithm did not separate data at all, as in Contraceptive Method Choice and Indian Liver patient. The cohesion values of the clusters were relatively low; low cohesion and no separation for Indian Liver Patient was perplexing if taking into account the cohesion and separation rates produced for this same data set from other algorithms. In regards to separation, the values were high for most data sets, indicating the clusters formed did not overlap.

6. Conclusion

Almost every clustering algorithm will find clusters in a data set, even if that data set has no natural clustering structure (Tan et al., 2006). This was observed in this experiment, as each algorithm was successfully able to generate clusters given different data sets, regardless of the data set characteristics. Each algorithm tested had different clustering tendencies for different types of data, so conclusions could be made across the algorithms, although the No Free Lunch Theorem holds for this project. The clustering tendencies was observed as each algorithm generated a different number of clusters with varying cohesion and separation for the same data sets. Without previous knowledge of the data sets, there was no measure on how well each algorithm could have clustered the set based on our parameters. Additionally, meaningful conclusions about the cohesion and separation were difficult to make given no comparisons could be made since the algorithms are not comparable and several tests per data set per algorithm was not run with varying parameter settings.

Future Works

In the future, to produce better and more meaningful results for different clustering algorithms, we would focus on running more tests with varying parameters on the same data sets. We could have spent more time tuning parameters for each of the five implemented algorithms. However, after analyzing the test results, it seemed that comparing results with varying parameters would have given us a basis for drawing conclusions for the clustering tendencies of the algorithms. Plus, implementing consistent maximum number of iterations would allow us to determine the convergence rates of the five clustering algorithms, making analysis more interesting. We realized there were several things we could have added to the experimental approach for better analyses.

References

D.W. Aha. UCI machine learning repository: Tic-tac-toe endgame data set, 1996. URL `http://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame`.

K. Bache and M. Lichman. Uci machine learning repository, 2015. URL *http : //archive.ics.uci.edu/ml*.

Anand M. Baswade and Prakash S. Nalwade. Selection of initial centroids for k-means algorithm. *International Journal of Computer Science and Mobile Computing*, (7):161–163, 2013.

M. Bohanec. UCI machine learning repository: Car evaluation data set, 1997. URL `http://archive.ics.uci.edu/ml/datasets/Car+Evaluation`.

Margarida G. M. S. Cardoso. UCI machine learning repository: Wholesales customers data set, 2014. URL `http://archive.ics.uci.edu/ml/datasets/Wholesale+customers`.

Li-Yeh Chuang, Yu-Da Lin, and Cheng-Hong Yang. An improved particle swarm optimization for data clustering. In *International MultiConference of Engineers and Computer Scientists, Hong Kong*, volume 1, 2012.

Inc. Dell. Data mining techniques, 2015. URL *http : //documents.software.dell.com/Statistics/Textbook/Data − Mining − Techniques*.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

J Handl, J Knowles, and M Dorigo. Ant-based clustering: a comparative study of its relative performance with respect to k-means, average link and id-som. In *Proceedings of the Third International Conference on Hybrid Intelligent Systems, IOS Press*, 2003.

Hugo. K-means clustering, n.d.

B. Johnson. UCI machine learning repository: Wilt data set, 2013. URL https://archive.ics.uci.edu/ml/datasets/Wilt.

Hamdi Tolga Kahraman, Ilhami Colak, and Seref Sagiroglu. UCI machine learning repository: User knowledge modeling data set, 2009. URL http://archive.ics.uci.edu/ml/datasets/user+knowledge+modeling.

T. Lim. UCI machine learning repository: Contraceptive method choice data set, 1997. URL http://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice.

J. Sander M. Ester, H.P. Kriegel and Xu. Dbscan-density-based spatial clustering of applications with noise.

David MacKay. An example inference task: clustering. *Information Theory, Inference and Learning Algorithms*, pages 284–292, 2003.

J Mishra. Comparing clustering algorithms.

K. Nakai. UCI machine learning repository: Yeast data set, 1996. URL http://archive.ics.uci.edu/ml/datasets/Yeast.

Prism. L16: Competitive learning.

Surendra Parsad Badu M. Venkateswarlu N. Ramana, B. UCI machine learning repository: Ilpd (indian liver patient dataset) data set, 2012. URL https://archive.ics.uci.edu/ml/datasets/ILPD+(Indian+Liver+Patient+Dataset).

Margaret Rouse. Data mining definition, 2008. URL *http* : *//searchsqlserver.techtarget.com/definition/data − mining*.

John Sheppard. Lecture notes in machine learning: Soft computing, 2015a.

John Sheppard. Csci 447 machine learning: Soft computing project #4 pdf, 2015b.

R. S. Siegler. UCI machine learning repository: Balance scale data set, 2014. URL http://archive.ics.uci.edu/ml/datasets/Balance+Scale.

Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Pearson, 2006.

I-Cheng Yeh. UCI machine learning repository: Blood transfusion service center data set, 2008. URL http://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center.