Machine Learning Project 4 Paper

Matthew Cirillo
Jake Danczyk
Henry King

Abstract

Clustering data is a challenging unsupervised machine learning task. In this paper we compared the effectiveness of 5 different clustering algorithms representing 3 styles of clustering on 5 interesting data sets. Using cluster cohesion and separation as an evaluation measure, DBSCAN was the overall highest performing algorithm. PSO and LVQ-I both had top performance on a single dataset but generally lagged DBSCAN and outperformed k-means. On a number of datasets LVQ-I and k-means had very similar performances. Ant colony clustering was able to find loose clusters for each data set, but its performance was slow.

1. Problem Statement

Our task for this assignment is to evaluate the performance of swarm-based algorithms on a data mining task. The swarm based algorithms we are comparing will be global best PSO and minimal model ant clustering. We will be using two data mining algorithms, k-means and DBSCAN. We will implement a competitive learning neural network, LVQ-I, as an alternative "clustering" method. These three will serve as baselines to compare against particle swarm optimization and ant-colony clustering. We have selected five datasets from the UCI library with which to test our algorithms: Leaf, Wine, Glass, Seeds, and Iris. They are all of moderate size, with numbers of data points ranging between 178 and 340.

1.1 Description of Algorithms

We'll be using five clustering algorithms that represent three approaches to clustering. K-means and DBSCAN are analytical algorithms, clustering primarily through distance measurements. LVQ-I is a neural network that clusters through competitive learning. Ant colony and particle swarm optimization cluster using swarm intelligence methods.

1.1.1 K-means

K-means seeks to find *k* centroids that represent the center of a cluster within the data set. They are initially set randomly and updated by assigning each point to a center by proximity and then calculating the mean of each point in the cluster. If the mean is different than the current centroid, it replaces it and the algorithm repeats.

1.1.2 DBSCAN

DBSCAN divides the points into core, border and noise groups. It operates on two parameters, *epsilon* and *min_pts*. A core point has at least *min_pts* points within *epsilon* distance from it. If a point is within *epsilon* from a core point, it is added to the cluster and its neighbors within *epsilon* are then added for consideration as well. Points that are not within epsilon of a core point are labeled noise.

1.1.3 LVQ-I

Learning vector quantization uses a single-layer neural network to match input vectors against the weights of the network. The output layer has an arbitrary number of neurons, each representing the center of a cluster of data. Training consists of each each output cluster "competing" for a given input by comparison to its weight vector. The unit whose weights are most similar is chosen and gets to claim the input as belonging to it's representative cluster. To keep one cluster from dominating amongst dense data a conscience factor is introduced to penalize outputs that repeatedly win each contest.

1.1.4 Ant Colony Clustering

A 2D grid is built and data vectors and ants are randomly distributed throughout. The ants move across the grid and have a probability of picking up and dropping items based on a measure of the local density of similarity of items. Like items are more likely to be dropped by other like items. Similarity is determined by the euclidean distance between data vectors.

1.1.5 g-best Particle Swarm Optimization

A swarm of particles moves through the n-dimensional search space. Each particle is a set of cluster centroids which shift positions each iteration. The position shifts are determined by a velocity vector, which has a random component and is also drawn towards both the best result achieved by the individual particle and the best achieved among all individuals.

1.2 Hypotheses

We will evaluate the performance of most the algorithms by measuring the mean cohesion and separation of the clusters. For ACC, we will display the 2D grid showing the final item positions colored by class. Cohesion represents the mean pairwise distances of points within a cluster and separation represents the mean pairwise distance between cluster centroids. We want our algorithms to return low cohesion and high separation scores, although this will vary by data set.

DBSCAN is expected to perform strongly on datasets with non-globular clusters. We predict K-means will be among the strongest on easy (well-separated) datasets which do have globular clusters. LVQ will perform similarly to K-means as each cluster centroid is incrementally pulled toward surrounding points.

We predict the relative performance of ACO and PSO will increase on the more difficult datasets, as we expect they will do a better job of exploring the entire search space.

2. Experimental Approach

The first step is determining the tunable parameters for each algorithm. This is highly dependent on the data set, but we will start from these values.

K-means	K = Num_classes
DBSCAN	NumPts = 2, Radius = 1/Num_Instances
ACO	Num_Agents = 3*Num_dimensions
PSO	10 particles, Num_centroids = 5*num_classes, inertia = 0.0001, c1 = 0.0001, c2 = 0.001, 50 iterations
LVQ-I	Num_outputs = Num_classes,Eta = 0.5

K-means is heavily dependent on the initial center points, they will be randomly selected and the algorithm run one hundred times on each data set. We will then repeat for K+/- 1, observe if performance increased for either, and if so continue adjusting K in that direction until performance begins to decrease. DBSCAN will be adjusted based on the performance of the prior iteration. If the number of clusters produced is much greater than the number of classes, we will trial both increased neighborhood radius and decreased numPts. When number of clusters is less than the number of classes, the next iteration will be tuned in the opposite direction (decreased radius and increased numPts). The accuracy metrics will also be evaluated for each iteration, and when no improvements are made through tuning in either direction that “branch” of tuning will be completed. To update LVQ-I, we will start with a sensible learning rate and increment up or down. Sparse data will likely perform better with a lower learning rate, as it will keep the centroids from being “tugged” out of an optimal position by distant inputs. Conversely, dense data will likely need a higher learning rate as the centers will get “tangled” in denser data and may miss the opportunity to move to a better position. We will then trial both increased and decreased number of output nodes and repeat. For ACO, we will run one hundred trials, and then trial an increasing and decreasing the number of ants by ten percent and repeat. We will continue varying ant numbers if performance gains are achieved. The PSO tuning was initially focused on tuning the inertia and acceleration parameters. Once these were adequate to see steady improvement over the course of an optimization trial, it was observed that points tended to end all belonging to a single cluster. To combat this, the number of clusters was increased. A number of five time the number of classes within the dataset was settled on as a reasonable compromise of search quality and speed.

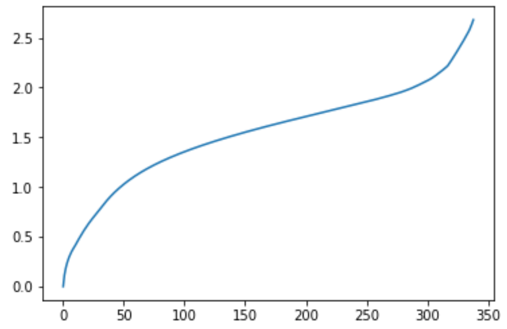
Once the algorithms are stable and optimized, we will run them on each data set and compare the cohesion and separation. For each data set, the algorithm with the best overall scores will be considered the strongest for that problem.

3. Results

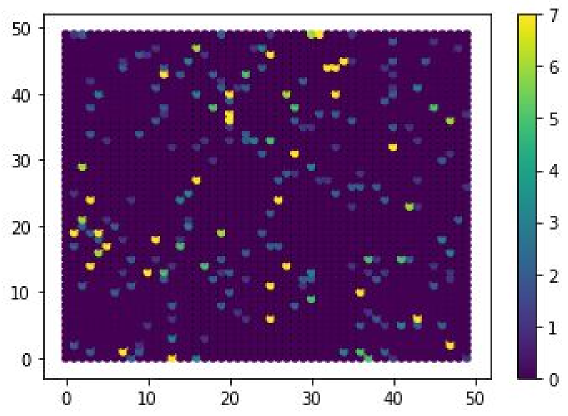
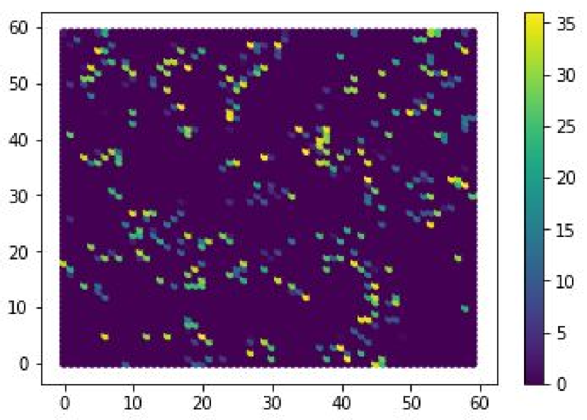
Here we display our results by data set. We provide a k-nn histogram that shows the mean distance between neighbors as k increases for each data set to contextualize our decisions. The histogram shows the mean distance between points in a cluster as k increases.

For each data set, we present the cohesion and separation scores for all algorithms but ant clustering. For ACC, we present the 2D grid that the ants traversed and the starting and final positions of the objects representing the data vectors. The colors represent the class of the data vector.

3.1 Leaf



Looking at the K-nn visual, we can see that there are clear elbows as k increases. This suggests that our clustering algorithms should find some number of dense clusters. This is a problem with a high number of classes, 36, and high dimensionality.

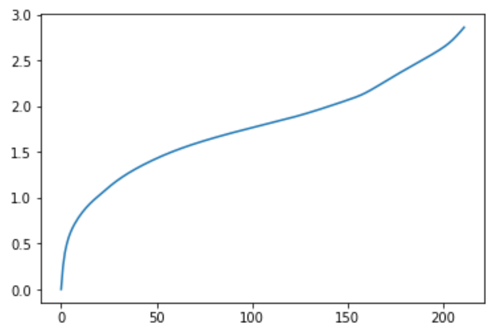



Algorithm	Cohesion	Separation	Cohesion/Separation
K-means	0.532	2.682	0.198
DBScan	0.974	5.759	0.169
LVQ-I	0.597	4.957	0.120
PSO	1.719	8.184	0.210

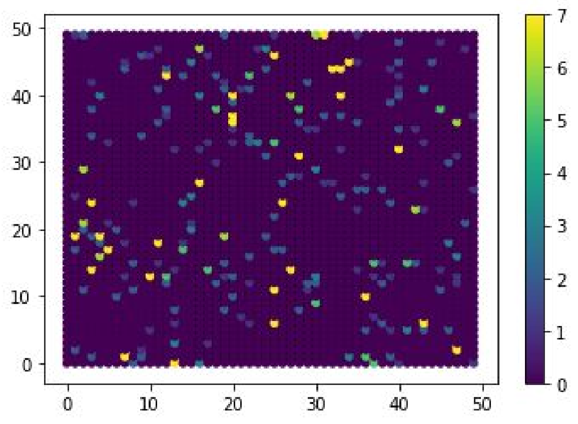
Ant clustering seemed to have trouble developing tight clusters on this set. This could be due to the complexity of the data set, where the high number of classes are not well differentiated. However, the clusters it did find appear to be of few classes. Most of the clusters appear to be made of items from the same class as well. PSO was the lagging performer. It's combination of poor cohesion but good separation will appear on other datasets. K-means and LVQ achieved relatively low cohesion when set up to find 30 clusters. LVQ scored the best on our combined metric because of its higher separation.

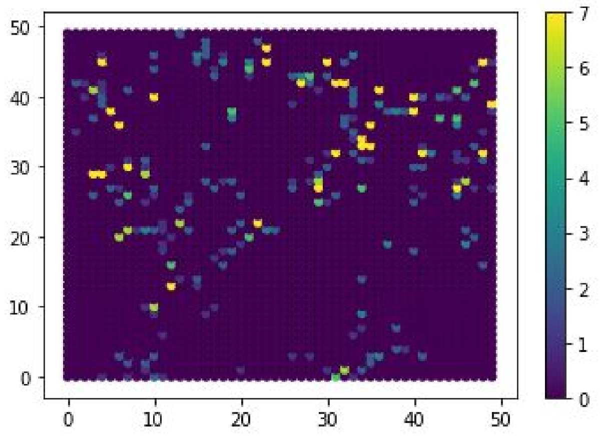
The poor performance of PSO and ACO on this dataset surprised us and did not support our hypothesis, as this was the dataset with the highest number of instances and dimensions. DBScan found non-radial clusters judging by the relatively high cohesion with respect to K-means and LVQ. The distance between these clusters was high as well.

3.2 Glass



Glass has a strong elbow and a weaker one. There should be some dense clusters in this set, but fewer than were in Leaf. This is a seven class problem and appears to be well separated.

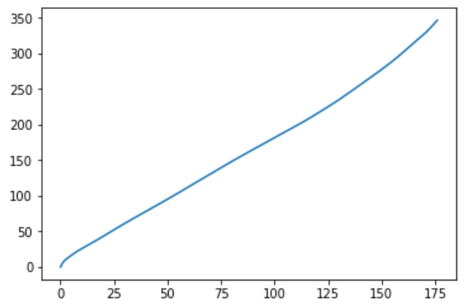




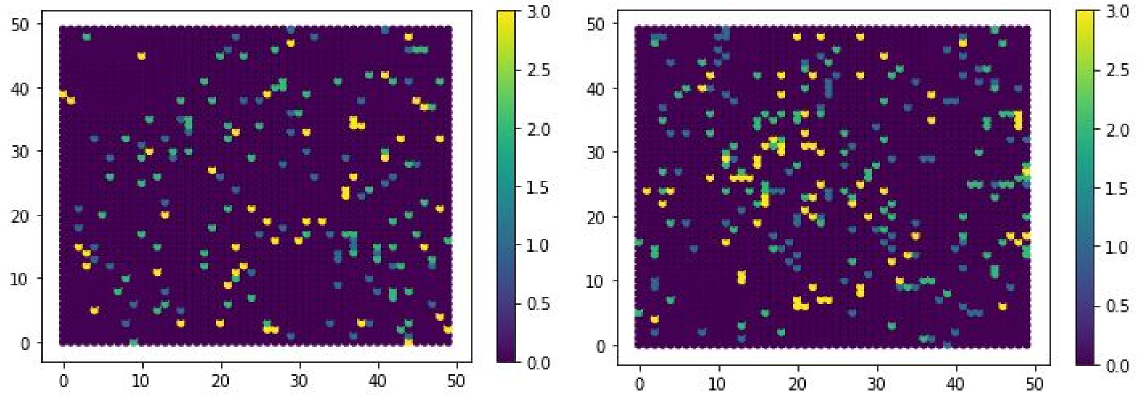
Algorithm	**Cohesion**	**Separation**	**Cohesion/Separation**
K-means	2.047	4.803	0.426
DBScan	0.945	5.861	0.161
LVQ-I	1.729	3.161	0.547
PSO	2.149	5.460	0.394

Ant clustering pushed the data into 3 loose clusters. According to the ants, there may be overlapping feature ranges between classes 5 and 6 that aren't immediately apparent and class 7's attributes are evenly distributed among the other classes. Two-dimensional scatter plotting of a number of attribute pairs indicated that this dataset did have non-globular clusters, which likely explains DBSCAN's strong performance, and provides some support for our hypotheses. The other three evaluated algorithms had much lower scores. PSO had the worst cohesion score, but was able to achieve relatively well separated clusters. LVQ-I had the opposite mix of scores, and was the worst overall.

3.3 Wine



Unlike the previous sets, Wine appears to not be a highly separated data set. This caused problems for some of our algorithms. Wine has 3 classes.

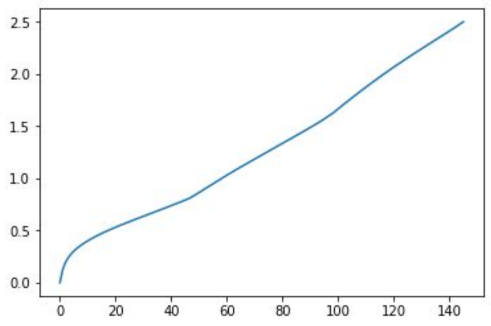


Algorithm	Cohesion	Separation	Cohesion/Separation
K-means	133.047	491.408	0.271
DBScan	90.714	465.491	0.195
LVQ-I	133.047	491.408	0.271
PSO	56.148	469.673	0.120

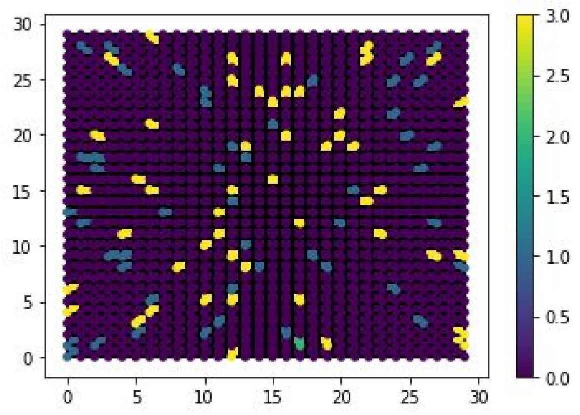
Ant clustering appears to have had trouble deciding where to put the items for this set. The final positions seem to suggest that the classes share similar attributes and aren't well distributed. There are a few dense regions of class 3 and 2, but most points are uniformly distributed. K-means and LVQ scored exactly the same. They clustered the same points when set to find 3 clusters. DBScan was the runner up to PSO which was able to find the densest clusters.

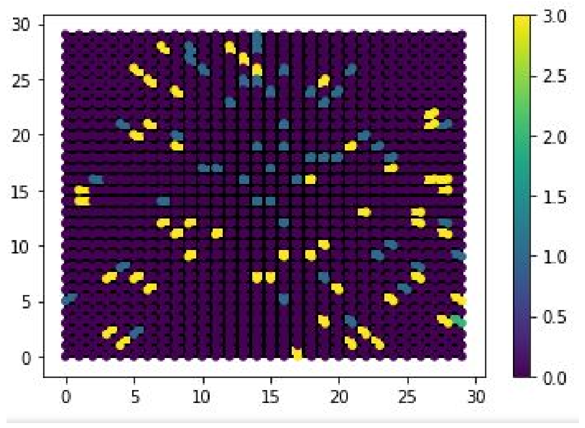
The strong performance of PSO on this relatively difficult dataset provides partial support for our hypotheses. The matching results here support our claim of similarity between LVQ and K-means.

3.4 Iris



Iris appears to have an elbow at the low end and the uniform separation as k increases. This is a famous dataset, with one class that is more distinct and two with a high degree of overlap.

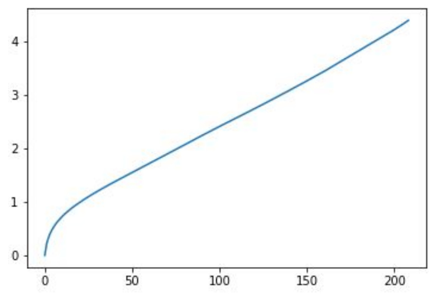




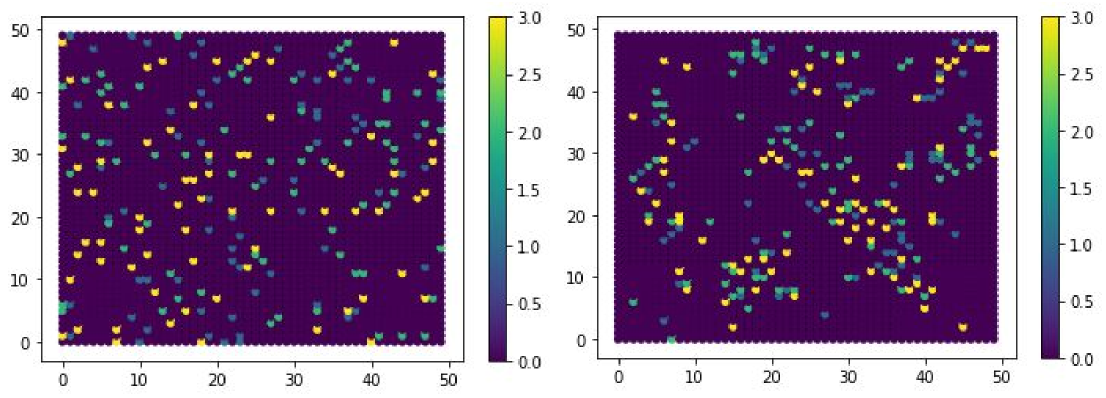
Algorithm	Cohesion	Separation	Cohesion/Separation
K-means	0.924	3.373	0.274
DBScan	0.677	3.585	0.189
LVQ-I	0.931	3.265	0.285
PSO	0.742	2.706	0.274

Ant clustering found a similarity between items of class 1 in this set, but didn't really separate out class 3. This might be indicative of the problem with ants where many small clusters are formed of items that should be in a larger one. K-means and LVQ performed similarly, again, creating clusters with nearly the same cohesion and average distance. PSO had a similar cohesion/separation ratio, but with lower values for both scores. DBSCAN was once again the top performer.

3.5 Seeds



Seeds appears similar to Iris, with an elbow at the low end, and then linearly increasing mean distances. It is a 3 class problem.



Algorithm	Cohesion	Separation	Cohesion/Separation
K-means	2.127	5.267	0.404
DBScan	1.506	4.931	0.305
LVQ-I	2.138	5.310	0.403
PSO	1.749	5.546	0.315

Ant clustering returned 7 clusters that aren't as uniform as its previous results. This could indicate that the classes in the data may have sub-groups within that have share distinct features with sub-groups from another class or that ant clustering might not be the best algorithm for this data set. K-means and LVQ shared similar results again for this test. DBScan performed well in this more homogenous data, achieving the lowest cohesion score. PSO was close behind DBSCAN, with worse cohesion but better separation.

4. Summary

According to our metrics DBSCAN was our top performer, finding asymmetric clusters in fairly homogenous data being one of its strengths. Using DBSCAN as an initial clustering method is a good idea in the future because it is fast and allows you to get a feel for the data. It may be determined after that another method is appropriate even though DBSCAN is quite flexible.

We hypothesized that K-means and LVQ are very similar and were correct. From the implementations we learned that both are based each point's distance to the closest centroid. The primary difference is that LVQ includes a bias against clusters that have claimed many points. Given the similar results a decision between the two would lead us to use K-means over LVQ as it was faster to return results.

If this work was extended, converting the ACO results to a set of clusters would be a priority to allow for numeric comparison with the other algorithms. Applying K-means or DBSCAN to the ant clusters would be possible ways to generate numeric evaluation measures.

References

1. Kruse R., Borgelt C., Braune C., Mostaghim S., Steinbrecher M. (2016) In: Computational Intelligence. Texts in Computer Science. Springer, London
2. Taher Niknam, Babak Amiri. An efficient hybrid approach based on PSO, ACO and k-means for cluster analysis. (2010) In Applied Soft Computing, Volume 10, Issue 1, 2010, Pages 183-197.
3. Christopher D. Manning,Prabhakar Raghavan,Hinrich Schütze. (2009) An Introduction Information Retrieval. Cambridge University Press. Cambridge, England
4. Aeberhard, S. (1991). Wine Data Set. Genoa, Italy, Institute of Pharmaceutical and Food Analysis and Technologies.
5. Silva P.F.B., Marçal A.R.S., da Silva R.M.A. (2013) Evaluation of Features for Leaf Discrimination. In: Kamel M., Campilho A. (eds) Image Analysis and Recognition. ICIAR 2013. Lecture Notes in Computer Science, vol 7950. Springer, Berlin, Heidelberg
6. B. German. (1987). Glass Identification Data Set. Aldermaston, Reading. Home Office of Forensic Science Service.
7. AP Engelbrecht. Computational Intelligence: An Introduction. Wiley. West Sussex, England. 2007
8. AP Engelbrecht, DW van der Merwe. Data Clustering using Particle Swarm Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 215-220, Canberra, Australia, 2003
9. Ester, M., Kriegel, H., Sander, J., A Density-Based Algorithm for Discovering Clusters. Institute for Computer Science, University of Munich, Munchen, Germany 1996
10. MaÅ‚gorzata Charytanowicz, Jerzy Niewczas, Seeds Data Set. Institute of Mathematics and Computer Science, The John Paul II Catholic University of Lublin. Lublin, Poland
11. R.A. Fisher. The Iris Data Set. UCI Machine Learning Repository. July, 1988