Genetic Clustering Algorithms vs Min K Means

Silhouette CGA

Jacob Hopkins
Computer Science
St Cloud State University
St Cloud MN USA
jrhopkins@stcloudstate.edu

ABSTRACT

Cluster Analysis is the job of partitioning objects into groups. It is the main task in data mining and often used in statistical analysis. It can be used in pattern recognition, image analysis, data compression, and machine learning.

Two of the more common approaches to clustering is K nearest neighbor and K means. K means is used to compare its performance to that of genetic clustering. K means is a sophisticated algorithm. It includes assigning k number of clusters to randomly selected points. Assigning points to the nearest cluster center. Relocating cluster centers to the mean of its assigned points. And repeating those two steps until convergence. It has strengths and weaknesses including occasional poor random starts.

Genetic algorithms are robust algorithms good at finding reasonable solutions often quickly in large search spaces. And unlike many traditional AI they do not break when the data is noisy or when the input changes slightly. This algorithm includes encoding the state space of the problem into a chromosome. A string that represents the states for all features. Repeatedly measuring performance called fitness. And through selection of better chromosomes, mutation of components of chromosomes upon slight chance, and crossover of two selected chromosomes to produce children, find solutions in repeated generations.

CCS CONCEPTS

Cluster Analysis, K Means, Genetic Algorithms

KEYWORDS

Cluster, Cluster Analysis, partitioning, data mining, machine learning, algorithm, K means, K nearest neighbor, Genetic Algorithms, AI, chromosome, crossover, mutation, selection, fitness, state space, convergence

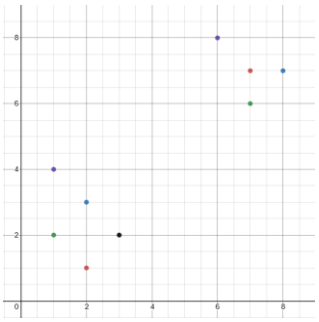
ACM Reference format:

Jacob Hopkins 2020. Three NP Complete Problems: A Brief Introduction

Cluster Analysis

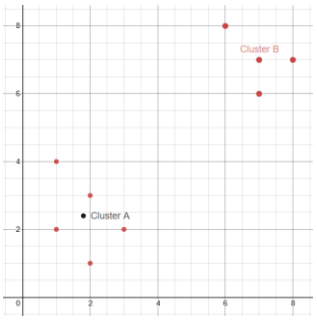
The type of cluster analysis explored is MINIMUM k-CLUSTERING. This includes a set of x and y pairs and the distance for each pair. For each of those pairs in the set that satisfy the triangle inequality z <= x + y, or for zero area triangle, vectors: |x+y| <= +|x| + |y|. partition those points into k number of disjoint subsets C1, C2, …., Ck so as to minimize the maximum distance between any two elements of the same subset. They must both be pairs on the plain, and the distances between points I the Euclidean Distances.

Suppose we had the set: [[2,1], [2,3], [1,2], [1,4], [3,2], [7,7], [8,7], [7,6], [6,8]]



The graph would look like the figure to the right: One can see with their eye that what appears to be two groups of points. One to the top right of the graph, and one near the origin.

A proper clustering algorithm could locate a central point of those two clusters and assign the corresponding clusters to the corresponding points. Some clustering situations are more complex than this. And some specific algorithms can struggle to understand the shape of the clusters as well.



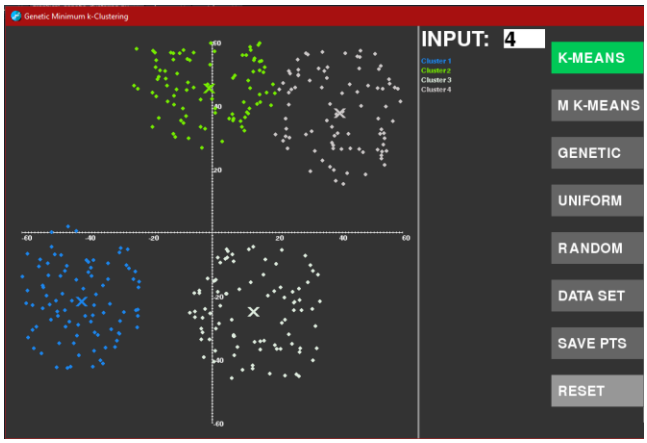
The two cluster centers of this example would be: [[1.8,2.4], [7,7]].

K Means

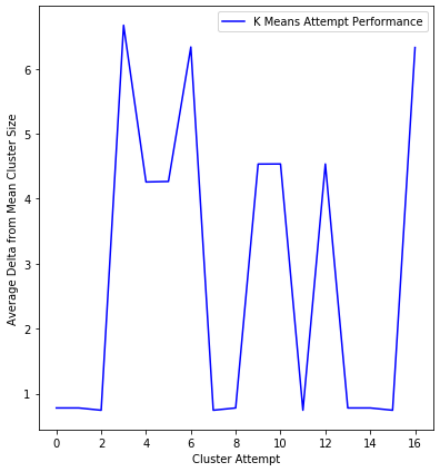
This popular machine learning technique is often effective on data but can struggle uncertain circumstance. When the data is not uniform blobs it cannot overcome not including

those nearest due to a close resemblance to K nearest neighbor. It can also struggle when it receives poor starts. To overcome this, I ran the k means algorithm 17 times, because of arbitrary favoritism and satisfactory results, and created my own heuristic for selecting those cluster centers which had better results. By comparing the difference between the average cluster size and the cluster size of each cluster and by averaging those scores for each run and taking the best, I found the algorithm was capable of clustering data:
Here is the K means results from clustering good blobs:

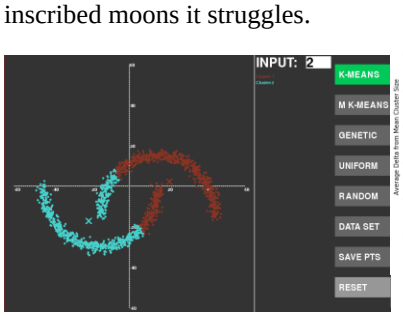
```
Time to compute K Means: 0.2465 seconds
Success Rate of K Means: 99.44444444444444 %
```

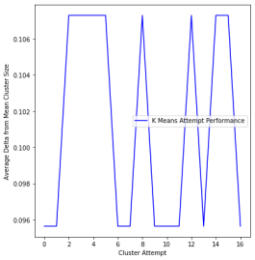


Here is the data resulting two the average delta from mean for each cluster:



By looking for those runs with the lowest values, attempts 2, 7, 11, 13, 15. We can select those which make the best cluster. However, when the data is inscribed circles, or inscribed moons it struggles.



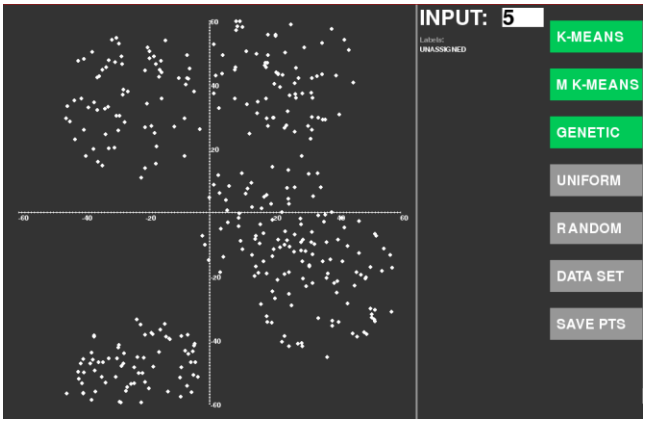



```
Time to compute K Means: 1.9450 seconds
Success Rate of K Means: 49.4 %
```

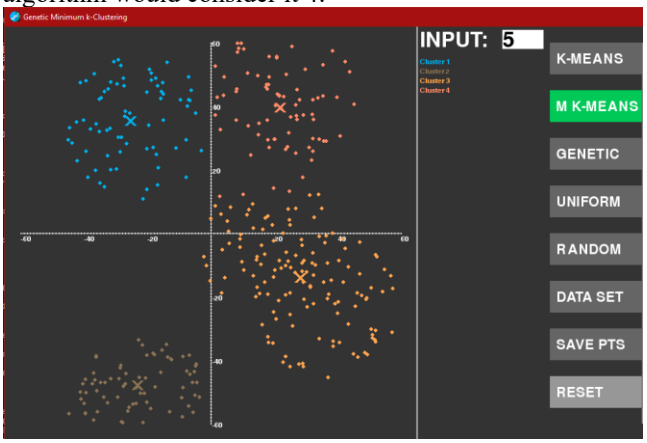
Min K Means

The ‘sklearn’ data sets and libraries are a more robust K means algorithm and testing data. By comparing the silhouette scores of each value of k I was able to select the best K value for a problem without needing input. The silhouette score is found by taking the distance of each cluster value to the cluster center and comparing that to the distance to those points of other clusters. Giving a ratio of how clumped or separated the data is.

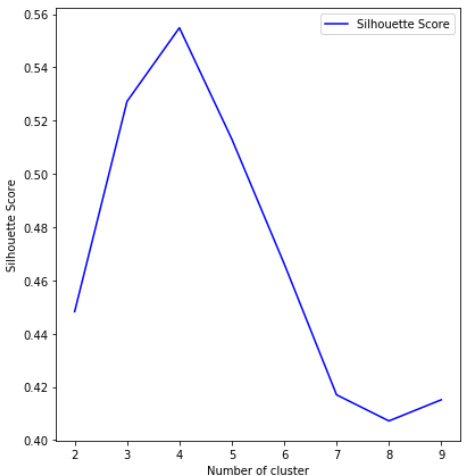
```
Time to compute min K Means: 0.3084 seconds
```



This data could be clumped into 4 or 5 clumps. But without prior knowledge about the data set the min k means algorithm would consider it 4:



That is because the algorithm is trying to maximize the silhouette score:



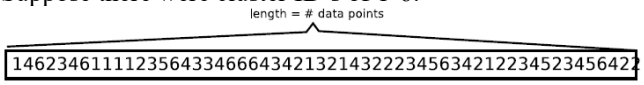
$$s(\vec{x}) = \frac{b(\vec{x}) - a(\vec{x})}{max\{b(\vec{x}), a(\vec{x})\}}$$

Above is the equation of the silhouette score. Scores between 0 and 1 indicates more clustered data, and the closer to 1 the more clustered. Negative values indicate clusters who are not in a cluster.

Genetic Clustering Algorithms

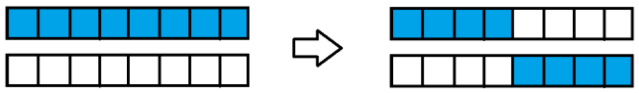
Chromosomal representation was done through a parallel array similar to Hruschka, Campelo, and Castro. For each point a integer that represents the ID of a cluster will be stored. This has high space requirements but leeway's easily into most any type of genetic operations available.

Suppose there were cluster ID's of 1-6:



Evaluation of chromosomes is done by using the points and the chromosome labels to find the silhouette score. Described above this will give an evaluation of -1 to 1. With negative meaning poorly performing chromosomes, and 1 suggesting positively performing chromosomes.

Mid-point crossover was used. This entails taking two parents from the population and creating two new chromosomes by exchanging the first and second half with the other parent.



Bias one-point mutation was used. This includes producing a random vector and randomly selected an allele in the chromosome to apply the vector to.

$$C(new) = C(old) + c$$

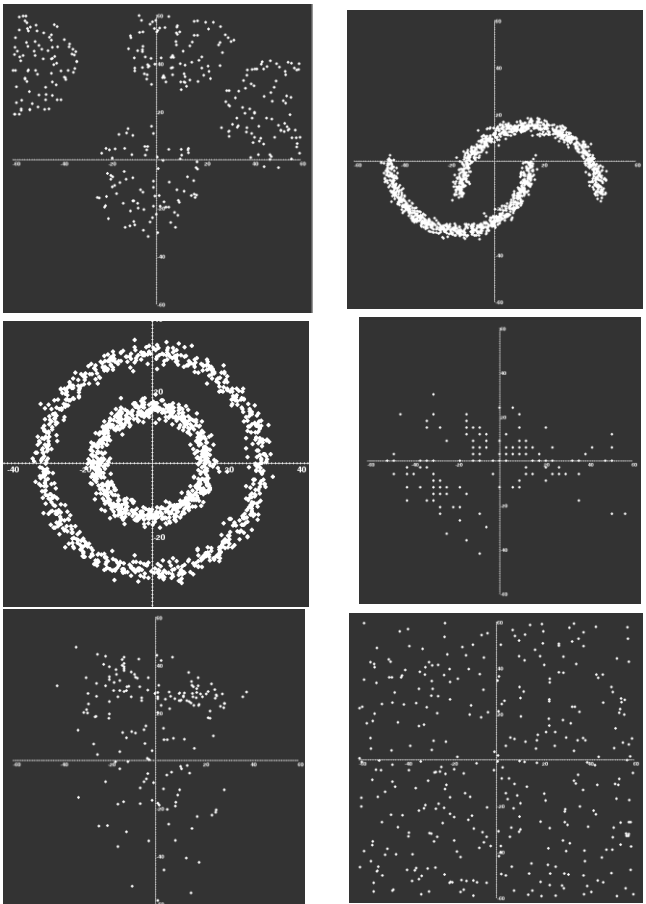
$$c = random +/- vector$$

After random initialization and evaluation for fitness. It runs selection, crossover, and mutation and re-evaluates fitness. The algorithm runs until convergence above 0 in silhouette score. This ensures that the data is somewhat clustered and no longer improving upon the data. The CGA has a population of 100 and replenishes 50% of the population each generation through selection. Mutation happens at 0.05% rate. It is otherwise a rather standard genetic algorithm, besides that with a Boolean value is can represent the highest fitness generation graphically and live; however, that comes with obvious slowdowns.

Selection is done with a chance for breeding a random parent with that of a high fitness parent. A random chance for breeding two random parents, and a random chance of breeding with a mutated offspring. Elitism is used keeping the 10% best of the last generation and replacing the other 50% keeping a random 40% of the last population.

PROBLEM INSTANCES

Sklearn is a python library with lots of data sets. An algorithm to create blobs randomly with overlap allowance was used as well. A graphical driver application was made to facilitate running and representing problems and solutions even as they are being solved. Points can be saved whenever an algorithm is not running. And Reset does not clear the points but does allow a second trail or a different trail to be run allowing data to be collected on the effectiveness of the algorithms. From sklearn 4 data sets are used: Moon and Circles are generated semi randomly and used to stress K means and compare to see if the CGA can overcome. Iris and Wine data sets are used. Both are 3 cluster data sets with through overlap. This can test the silhouette function for its ability to identify clusters that are not tailored for k means like solutions. Labels are provided from all data sets allowing a rough success rate to be found. Pictures of these data sets and a uniform distribution:



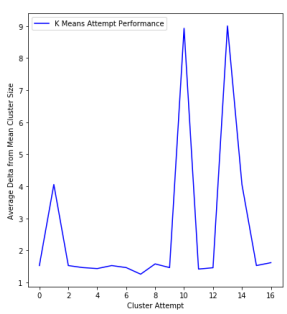
Testing Procedure

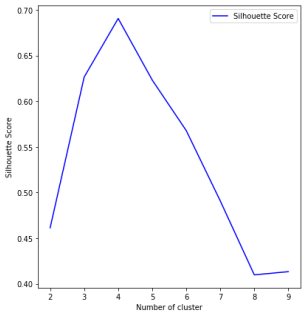
Results from my application are printed into console. One can launch this application and repeated run the desired algorithm on the desired data set, but this can be inefficient. Instead a main driver program was used to run many trials upon many different examples of the different datasets (besides uniform, wine, and iris) to compare performance upon random chance examples on good and bad sets. The genetic algorithm and K means were run 20 times for each set and averaged for performance.

Performance

Blobs (4 total)

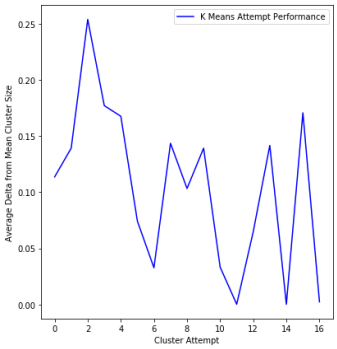
Algorithm	Run Time	Rough Accuracy
K Means input k = 4	0.3475 s	95%
Min K Means k = 4	0.2798 s	94%
CGA k = 4	0.4861 s	94%

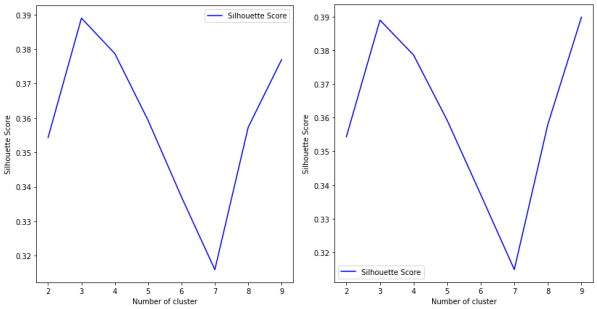




Circles (2 total)

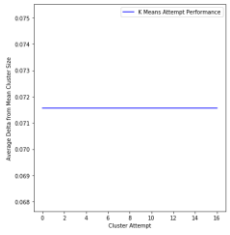
Algorithm	Run Time	Rough Accuracy
K Means input k = 2	1.974s	49.6%
Min K Means k =3/9	0.8124s	34.6% @ k=3 7.8-16% @ k = 9
CGA k = 3	3.4581s	35.6%

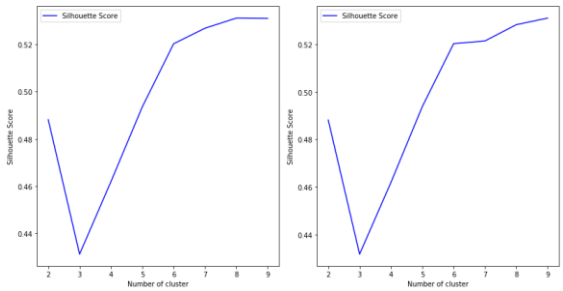




Moons

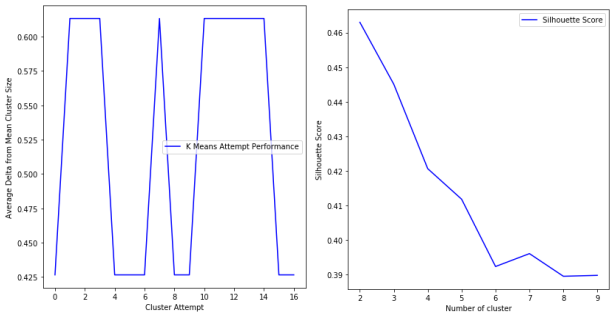
Algorithm	Run Time	Rough Accuracy
K Means input k = 2	1.4532s	48.2%
Min K Means k =8/9	0.7952s	46%
CGA	2.834	46.2%





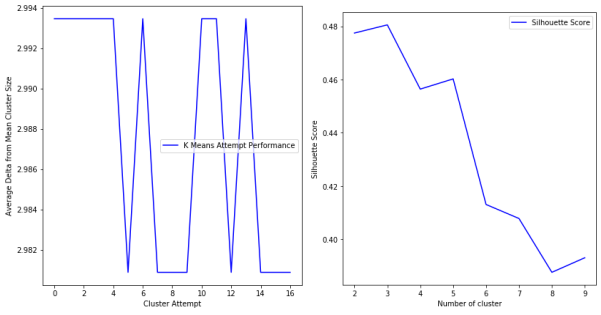
Iris

Algorithm	Run Time	Rough Accuracy
K Means input k = 3	0.1321s	98%
Min K Means k =2	0.2424s	44.6%
CGA k = 2	0.4567s	46%



Wine

Algorithm	Run Time	Rough Accuracy
K Means input k = 3	0.9896 s	88.2%
Min K Means k =3	0.2667 s	96.6%
CGA k=3	1.4583s	96.3%



Changes

Exploration into the idea of fission or fusion for mutation is interesting. By changing a random allele in the chromosome to that different or the same of a nearby alleles would be a more interesting mutation type and may or may not outperform bias one-point mutation. Exploration into larger data sets and k cluster counts would also be an interesting comparison or this is low k values.

CONCLUSIONS

The GCA could definitely be improved in terms of speed if I used generators instead of lists in python, but I think this served well. The results often were similar, so possibly using inertia instead of silhouette, or better applying either could get better results

ACKNOWLEDGMENTS

Thank you to the SCSU Computer science department for the opportunity to create a paper in this format to get the experience of scientific papers.

REFERENCES

[1] Pulkit Sharma 2019 'The Most Comprehensive Guide to K-Means Clustering You'll Ever Need' https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/

[2] Chris Piech 2013 'K Means' https://stanford.edu/~cpiech/cs221/handouts/kmeans.html

[3] Eduardo R Hruschka & Nelson Ebecken 2003 'A genetic algorithm for cluster analysis' https://www.researchgate.net/publication/220571471_A_genetic_algorithm_for_cluster_analysis

[4] Marco Painho and Fernando Bação 2000 'Using Genetic Algorithms in Clustering Problems' http://www.geocomputation.org/2000/GC015/Gc015.htm

[5] Imad Dabbura 2018 'K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks' https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

[6] M.C. Cowgill, R.J. Harvey, L.T. Watson 1999 'A genetic algorithm approach to cluster analysis' https://www.sciencedirect.com/science/article/pii/S0898122199000905

[7] Minitab 2019 'Interpret all statistics and graphs for Cluster K-Means' https://support.minitab.com/en-us/minitab/18/help-and-how-to/modeling-statistics/multivariate/how-to/cluster-k-means/interpret-the-results/all-statistics-and-graphs/