Machine Learning Project 4: Final Paper

Brett Layman

Kirby Overman

Carsen Ball

Abstract

In this paper, we compare the performance of five different clustering algorithms, K-Means, DB-Scan, a competitive learning neural network, Particle Swarm Optimization, and Ant Colony Optimization in their ability to effectively cluster five different data sets. We compared our algorithms using a silhouette fitness function. Through experimentation, we found that DB-Scan was typically the best clustering algorithm across all five datasets.

1. Introduction

This section provides a brief overview of the project.

1.1 Algorithms Used

We implemented five algorithms to cluster 5 data sets. Below is a brief description of the algorithms

1.1.1 K-Means

K-Means clusters the data to the amount of clusters specified by the user by solving for K centers, where at the end of the algorithm each data point will be assigned to the nearest center by using Euclidean Distance. This is done by randomly choosing K centers from the data set, then all points are assigned to the smallest euclidean distance to the center. Once the initial assignment has been done, the centers are recalculated based off of the average value of the data points within each cluster. The points are reassigned to the new centers, and the process continues until no change is seen in the value of the centers or in the data points assigned to the centers.

1.1.2 DB-Scan

DB scan clusters data based of off density. The algorithm choses a point at random, then checks to see if that point has the user specified number of minimum points within the distance epsilon, user specified distance, of the point. Euclidean distance was used as the distance measurement. If the point has enough minimum points within epsilon, then the point becomes a core point. The point, and all of the points within the range are given a cluster label. All of the points that were assigned to the cluster are then checked to see if they have enough points within epsilon to become core points themselves. If they do, all of the points that were within epsilon then become part of the same cluster, and are checked as to be core points as well. When there are no more points to be left to check as core points, the algorithm choses another point at random, and if it has not been classified the process repeats to create a new cluster. If a point was not added to a cluster, and did not have enough points around it to be a core point, it is classified as noise.

1.1.3 Competitive Learning Neural Network

The competitive learning neural network has one hidden layer. The number of nodes in the hidden layer correspond to the number of attributes that the data points in the data set have. The user selects the learning rate, and the number of output nodes. Each data point is fed into the network and for each output node $\mathbb{W} \cdot \mathbb{X}$ is calculated. The output node with the largest sum of $w_i \cdot x_j$ assigns its cluster value to the point. Because the weights and inputs are normalized $\mathbb{W}+ = \eta \mathbb{W} \cdot \mathbb{X}$. The process continues until all points have been fed through the network.

1.1.4 Particle Swarm Optimization

Our PSO algorithm is based off of an algorithm proposed by Cura [5]. The main difference for our algorithm is our optimization objective, which uses Silhouette coefficients instead on a basic distance to centroid measure. This made our algorithm much slower, but also allowed for comparison with algorithms like DBScan and ACO. Each particle represents a set of clusters, and thus a potential solution to the clustering problem. We use a basic PSO particle update equation where velocity is calculated using personal best and global best terms:

$$v_k^{t+1} = v_k^t + c_1 \cdot r_1(z_g - z_k^t) + c_2 \cdot r_2(z_p - z_k^t)$$

Where z_k^t is the current position of the particle, z_g is the current global best, and z_p is the current personal best. c_1 and c_2 are scalars that determine the relative influences of the global and personal best vectors. r_1 and r_2 are random vectors with the range [0,.1] for each value.

Our PSO algorithm uses the K-Means algorithm to re-cluster points between each particle update. This has the effect of re-centering the clusters around average distances.

1.1.5 Ant Colony Optimization

For ACO we decided to use the pheromone based clustering approach instead of the more common cemetery based approach. With this approach we were able to control for how many clusters we wished to use. For our pheromone update equation we used

$$\tau_{i,j}(t+1) = (1-\rho)\tau_{i,j}(t) + \rho \Delta \tau_{i,j}^{best}$$

where $\Delta \tau_{i,j}^{best} = \frac{1}{F_{best}}$. Here, F_{best} is the highest fitness. For ACO, we used a different fitness function from the silhouette that we used for the other four algorithms. The fitness function we chose was the following:

$$minF(\mu) = \sum_{j=1}^{K} \sum_{i=1}^{N} \sum_{v=1}^{n} w_{ij} ||x_{iv} - \mu_{iv}||^2$$

where K is the number of clusters, N is the size of the dataset, and n is the length of each data vector. Since this is a minimization problem with ACO, lower fitness measures are better.

1.2 Hypothesis

We expect both ACCA and PSO to produce clusters with a higher, more optimal fitness than the traditional data mining methods; meaning that that the data points in a cluster will be in closer average proximity to the centroids. This hypothesis is based on the assumption that these algorithms are less likely to not get stuck in a local optima, because of their more comprehensive exploration of the search space. For instance, our PSO algorithm explores a variety of initial centroid positions, one for each particle. Whereas K-Means chooses only one set of initial centroid positions. That said, we expected PSO to be the most computationally costly, because it has to basically run k-means on every particle in addition to calculating velocity, so it may run into some time limitations in terms of the number of iterations we can run. K-means should also be relatively costly, because of all of the distance calculations it has to make. ACCO, DB-Scan, and the neural network should run with a lesser computational cost than K-means or PSO. We expect the neural network to converge the fastest, because of it's relatively simple learning rule. However, we think the simplicity of this rule may lead to worse clustering performance than the other algorithms.

For convergence rates, given that K-Means is NP-Hard, we expect it to typically take the longest, especially with our larger datasets. Since ACO and PSO are optimization heuristics for NP-Hard problems, we expect them to be about as fast as our competitive learning algorithm. That being said, given our experience with training neural nets, we expect the three of these algorithms to be fairly slow — possibly 15-20 minutes per execution on a basic personal computer. Given that DB-Scan has a worse case time complexity of $O(n^2)$, we believe that it will be by far the fastest.

2. Methods

Discussed below are the methods that were used when running the algorithms.

2.1 Algorithm Analysis

To analyze the performance of a clustering algorithm we used a silhouette fitness function. Silhouette coefficients produce a number in the range: [-1,1], where higher values represent more accurate clustering. The coefficient is calculated as follows: a(i) represents the average distance from i to all other points within i's cluster, and b(i) represents the average distance from point i to all of the points in the cluster nearest to i's cluster (i.e. with the smallest average distance to the points in that cluster from i). Let s(i) be the silhouette coefficient of i:

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

If n is the number of clusters and n_c is the number of data points within a cluster then the fitness of a clustered data set can be calculated as the average silhouette coefficient:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{n_c}\sum_{k=1}^{n_c} s(i)$$

We ran each algorithm on five UCI Repository data sets and calculated the fitness of the clusters produced. The data sets are as follow, Seeds Data Set, Energy Data Set, Shuttle

Stat Log Data set, Pima Indians Diabetes Set set, and HTRU2 Data set. We compare the average fitnesses produced by each algorithm for a data set.

2.2 K-Means

The only tunable parameter for K-Means is K, the number of clusters desired. To determine K, we used the number of clusters produced by DBScan, where the silhouette fitness produced by DBScan was high, and the percentage of data points that DBScan classified as noise was low, and the number of clusters was meaningful. For example, 100 clusters for a data set with 200 points might not produce any meaningful clusters in the data. We ran K-Means 5 times for each different data set, for the different K values. For the Seeds data set we ran K-Means for K = 4, 16. For the Energy data set K = 2, 5 were used. For the Shuttle data set K = 6, 12 were used. For both the Indian Diabetes data set and the HTRU2 data set k = 2,4 were used.

2.3 DB-Scan

The tunable parameters for DB-Scan are the minimum points that have to be within an epsilon distance of a potential core point, epsilon being tunable as well. The data was normalized between 0 and 1, so the epsilon values of 0.7, .5, .3, .2, .1, .05 were used. Because no free lunch applies, a wide range of epsilon values were used. After running experiments, it was noticed that epsilon values less than .05 were producing non meaningful results. High epsilon values were kept to analyze the result when the minimum point values were large as well. The minimum number of points used to produce meaningful results varied based on how many data points there were in the data set. We multiplied the size of the data set by 0.25, 0.15, 0.1, 0.07, 0.05, 0.03, 0.01 to calculate the minimum number of points used. For each data set, DB-Scan was run for each pair of minimum point and epsilon values.

Seeds	Energy	Shuttle	Indian Diabetes	HTRU2
52	192	250	192	250
32	115	150	115	150
21	77	100	77	100
15	54	70	54	70
10	38	30	38	30
6	23	50	23	50
2	8	10	8	10

Table 1: Minimum Point Parameters used Per Data Set for DB-Scan

2.4 Competitive Learning Neural Network

The tunable parameters for the network were learning rate and the number of output nodes. To produce results that would be meaningful to compare to PSO and K-Means, the K values used for K-Means were used for the number of output nodes, and .5, .05, .01, .005, .001 were used for learning rates. Because no free lunch applies, a wide range of learning rates were used. For each data set the network was run 5 times for each K, learning rate pair.

2.5 Particle Swarm Optimization

For PSO, we tuned the global and personal best scalars, c_1 and c_2. Originally we were hoping to tune the population size as well, but the algorithm became to slow for our time constraints for larger populations, so we kept a population size of 20. We used three test cases: high c_1 low c_2 (values 8 and 4), low c_1 high c_2 (values 4 and 8), and equal c_1 and c_2 (8 each). Testing along these dimensions allowed us to modify how elitist the algorithm is, in terms of tending towards the current best global solution.

2.6 Ant Colony Optimization

For ACO, we experimented with the numerous parameters until we found sufficiently effective parameters. These resulted on a ρ value of 0.1, and 10 ant per all datasets. For the HTRU2 dataset, we ended with 25 epochs until convergence, 200 epochs for the seeds dataset, and 100 epochs for the rest. For the initialization of pheromones, we randomly selected from a wide range of 0.3 to 10. Despite this high range, ACO was still able to converge quickly.

3. Results

In this discussion the results are presented. Though multiple parameters were used for tuning the algorithms, the results presented are from the optimal parameters found. Table 2 shows the resulting two most optimal sets of parameters per data set for DB-Scan. The sets returned the best ratio of fitness to noise, and the resulting number of clusters were used for K-Means, the Competitive Network, and Particle Swarm Optimization. Table 3 shows the results of K-Means on the five data sets. Table 4 shows results of the Competitive Learning Neural Network.

Data Set	Number Of Clusters	Fitness	Percent Noise	Epsilon	Minimum Points
Seeds	16	0.56	10%	.2	2
Seeds	4	.47	36%	.3	15
Energy	2	0.55	11%	.7	38
Energy	5	0.368	18%	.5	23
Shuttle	6	0.68	4%	.2	10
Shuttle	12	0.64	15%	.1	10
Ind Diabetes	4	0.46	24%	.3	8
Ind Diabetes	2	0.42	9%	.5	23
HTRU2	4	0.67	11%	.2	10
HTRU2	2	0.63	4%	.5	30

Table 2: Fitness Produced By DB-Scan for the Optimal Parameters

Data Set	K	Run1	Run2	Run3	Run4	Run5
Seeds	16	0.315	0.293	0.297	0.304	0.315
Energy	5	0.258	0.262	0.257	0.272	0.321
Shuttle	6	0.354	0.488	0.442	0.354	0.467
Ind Diabetes	4	0.21	0.139	0.167	0.165	0.139
HTRU2	4	0.246	0.246	0.34	0.34	0.34

Table 3: Fitness Produced By K-Means

Data Set	Output Nodes	Learning Rate	Run1	Run2	Run3	Run4	Run5
Seeds	16	0.01	0.142	0.163	0.281	0.144	0.162
Energy	5	0.001	0.283	0.297	0.23	0.39	0.263
Shuttle	6	0.005	1.0	0.638	0.832	0.832	0.63
Ind Diabetes	4	0.01	0.226	0.193	0.688	0.354	0.124
HTRU2	4	0.01	0.506	0.511	0.44	0.485	0.325

Table 4: Fitness Produced By Competitive Learning Network Optimal Learning Rate

Data Set	K	Run1	Run2	Run3	Run4	Run5
Seeds	16	0.32	0.31	0.32	0.30	0.32
Energy	5	0.32	0.36	0.37	0.46	0.32
Shuttle	6	0.49	0.55	0.45	0.48	0.47
Ind Diabetes	4	0.29	0.21	0.17	0.24	0.25
HTRU2	4	0.39	0.36	0.35	0.37	0.34

Table 5: Fitness Produced By Particle Swarm Optimization

Data Set	K = 2	K = 4	K = 8	K = 12
Seeds	0.0047	0.0048	0.0048	0.0048
Energy	0.0091	0.0089	0.0087	0.0071
Shuttle	0.0048	0.0048	0.005	0.0047
Ind Diabetes	0.009	0.0089	0.0087	0.0084
HTRU2	0.0286	0.0273	0.0263	0.0231

Table 6: Fitness Produced By Ant Colony Optimization

Table 5 shows the results of Particle Swarm Optimization on the five data sets.
Table 6 shows the results of ACO on the five data sets.

3.1 Algorithm Comparison

Here we compare the average fitness of four algorithms for the 5 data sets. When applicable the K Values, which were the same as number of clusters in Db-Scan and the number of output nodes of the network, used per data set were K = 16 for seeds, K = 5 for energy, K = 6 for Shuttle and K = 4 for Indian Diabetes and HTRU2. Table 7 shows the average fitness of DB-Scan, K-Means, the Competitive Learning Network, and Particle Swarm optimization.

Data Set	DB-Scan	K-Means	Network	Particle Swarm
Seeds	0.56	0.305	0.180	0.315
Energy	0.368	0.274	0.292	0.36
Shuttle	0.68	0.421	0.79	0.49
Ind Diabetes	0.46	0.164	0.317	0.23
HTRU2	0.67	0.30	0.453	0.36

Table 7: Comparison Of Average Fitness

4. Discussion

In this section we discuss our results and hypotheses.

4.1 Algorithms

Here we discuss the results from the five algorithms.

4.1.1 DB-Scan

We chose to use DB-Scan to find the natural clusters within a data set, so comparison of the other algorithms would not be meaning full. Because of the decision to use DB-Scan to tune the other algorithms, the fitness of DB-Scan cannot be used to compare the performance of the algorithms. The algorithm was able to produce the most meaningful clusters when the minimum points were relatively small, but the epsilon value was largely dependent on the data set, and the density that existed within the data set.

4.1.2 K-Means

K-Means performed best on the shuttle data set with an average fitness of 0.421. The competitive neural network and DB-Scan were computationally less expensive, which is understandable because K-Means might iterate over the data set many times depending on how many iterations it takes for no change to be seen in the cluster centers and data points assigned to the clusters. K-Means ran for more iterations before convergence than DB-Scan.

4.1.3 Competitive Learning Neural Network

The competitive network performed best on the shuttle data set with an average fitness of 0.79. On average, it cannot be said that the network outperformed the other base line methods. The fitness was very dependent on the data set, but a learning rate less than or equal to 0.01 seemed to produce the best fitness.

4.1.4 Particle Swarm Optimization

As expected, PSO appeared to perform better than K-Means on each dataset, but not by a large amount. It's likely that the similarity in performance is due to the fact that PSO uses k-means during each iteration. It's likely that PSO had an advantage because it initializes a population of particles, and is able to choose the best of many initial cluster positions. It's also possible that the movement of the particles helped PSO to reach a more optimal solution, although it only was given 3 iterations to update the particle positions. We think that given more iterations, our PSO algorithm would have likely reached a better solution, although it is possible that it converged on a local minima after 3 iterations. We also think that having a large population would have helped, by exploring more of the solution space from the very beginning.

4.1.5 Ant Colony Optimization

ACO converged much quicker than expected. This is likely due to the fact that the implementation favored exploitation as opposed to exploration. We likely could have improved this results by considering the few best agents in the colony rather than the single best. Additionally, if we had introduced crossover as suggested in [6] to improved the exploitation of the best solutions. ACO also wasn't as accurate as expected. This is likely because of the fast convergence rate, but during trials, ACO produced larger initial fitness values (remember, for ACO we wanted to minimize). The larger overall fitness values may have caused the quick and significant differences in pheromone levels that prevented better exploration. ACO tended towards better performance with more clusters as one would expect due to the fitness measure summing over its clusters, i.e., more clusters implies smaller clusters which in of itself explains smaller sums.

4.2 Summary

We expected Particle Swarm Optimization to produce a higher average fitness than the baseline methods. DB-Scan outperformed the other methods, as was expected because we based the number of clusters off of the optimal result for DB-Scan. This was to decide on the number of clusters to use for the other algorithms, when applicable. Particle Swarm produced an average fitness of 0.315 for the Seeds data set, 0.36 for the Energy data set, out performing K-Means and the Network. For the Shuttle data set, the Network produced the highest average fitness of 0.79 and for the HTRU2 data set the Network produced a fitness of 0.45, and a fitness of 0.317 for the Indian Diabetes data set. For all cases the Network out performed K-Means and Particle Swarm Optimization. No free lunch applies for the algorithms and data sets used. The Particle Swarm algorithm expands on K-Means so it was not surprising that it out performed K-Means in all five cases. Given that we used a different fitness function for ACO, we had a difficult time comparing it to the rest of the algorithms in terms of accuracy. We do note; however, the unexpectedly fast convergence rate compared to PSO, which we expected to be approximately the same. Additionally, ACO did produce very small fitness measures (desired given it was posed as a minimization problem) we do believe that it performed fairly well, average at least across all datasets with respect to the other four algorithms.

References

[1] Miroslav Kubat (2015) *An Introduction to Machine Learning* Springer. Chapter 5, Artificial Neural Networks.

[2] UCI Machine Learning Repository (1987) archive.ics.uci.edu/ml/index.php

[3] Rudolf Kruse & Christian Borgelt & Christian Braune & Sanaz Mostaghim & Matthias Steinbrecher (2016) *Computational Intelligence* Springer. Second Edition.

[4] Urzula Boryczka (2008) *Ant Clustering Algorithm* Intelligent Information Systems. pg 337 - 386.

[5] Cura, T. (2012) *A particle swarm optimization approach to clustering.* Expert Systems with Applications, 39(1), 1582–1588.

[6] Bao-Jiang Zhao (2007) *An Ant Colony Clustering Algorithm* IEEE. 2007 International Conference on Machine Learning and Cybernetics.