CSCI 447 Project 4: Swarm-based Clustering Fall 2015

Brandon Fenton
Department of Mathematical Sciences
Montana State University
Bozeman, MT 59717-2400
Email: brandon.fenton@gmail.com

John Sherrill
Department of Mathematical Sciences
Montana State University
Bozeman, MT 59717-2400
Email: prof.sherrill@gmail.com

***Abstract*—Five seperate clustering algorithms were implemented in the interest of comparing convergence and clustering performance across 10 seperate datasets. Four of the five algorithms may be considered "swarm-based" methods as they use a population of candidate solutions that communally search for optimal solutions. Clustering performance was evaluated in terms of "quantization error". The algorithms were evaluated by comparing the distributions of quantization errors produced from multiple simulations across the 10 datasets. Since there were several algorithm, each with distinct parameters and multiple datasets tuning was performed manually on a per-example basis with initial reference values provided by the literature. The five algorithms implemented were: k-means, DB-scan, competitive learning, particle swarm optimization (PSO), and ant colony optimization (ACO). Of these five, it was hypothesized that DB-scan and ant colony optimization would yield the lowest quantization errors across all data sets as 1) neither makes broad assumptions about the problem and 2) the literature provided results seemed to suggest these were the most generally powerful classifiers. Results in fact show that the DBScan and k-means method performed the best. It is hypothesized that this is due to the fact that they are not tuned which is a critical part of achieving quality performance for this data mining task.**

I. Introduction

A common data mining task is organizing data into k seperate classes, i.e., clustering the data into seperate clusters. While a general definition of a cluster does not exist, for the purposes of this project a cluster is defined as a group of data points that are close to one another in terms of Euclidean distance.

Given a set of points $D = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ and a number of clusters, n_c, let a partition P of D be defined as a set of disjoints subsets of D such that the union of all said subsets is D. That is, a partition P of D is a set $\{P_i\}$ such that $\cap_i P_i = \emptyset$ and $\cup_i P_i = D$. Let each P_i have centroids

$$\mathbf{c_i} = \frac{1}{|P_i|} \sum_{\mathbf{x} \in P_i} \mathbf{x}$$

where $|P_i|$ denotes the number of elements in P_i. Define the "quantization error" of such a partioning as

$$QE(P) = \frac{\sum_{i=1}^{n_c} \frac{1}{|P_i|} \sum_{\mathbf{x} \in P_i} |\mathbf{x} - \mathbf{c_i}|}{n_c} \tag{1}$$

This is the average Euclidean distance between each point and it's respective partition, averaged across all partitions. For this project we seek a partitioning (henceforth referred to as a clustering) $P = \{P_i\}$ with $|P| = n_c$ minimizing the quantization error. This measure of performance was chosen as it appeared in the literature [3] and was thus a metric that enabled comparisons of results.

Five clustering algorithms were implemented: k-means clustering, ant colony optimization (ACO), particle swarm optimization (PSO), DB-Scan, and a competitive learning method (CL). The results from k-means, DB-Scan, and CL were considered baselines as k-means was previously implemented and is not a swarm-based method. The ACO and PSO results were then compared with the others for both individual comparisons and swarm-based/non-swarm-based comparisons.

II. Algorithm Descriptions

A. k-Means Clustering

Several variations of the k-means clustering algorithm have existed since at least 1957 [1]. The version used for this project is the most common and simplest except that the stopping criterion is based on the number of iteration of the algorithm and not a check for convergence.

Given that there are n_c clusters to be formed, n_c points were randomly chosen from the data set D to be cluster centroids $\mathbf{c_i} i \in \{1, \ldots, n_c\}$. Each point $\mathbf{x} \in D$ was then assigned to the cluster C_i containing the centroid closest to $\mathbf{x}$, i.e. $\mathbf{x} \in C_i$. Each centroid, $\mathbf{c_i}$ was then updated to be

$$\mathbf{c_i} = \frac{\sum_{\mathbf{x} \in C_i} \mathbf{x}}{|C_i|}. \tag{2}$$

Data points were then reassigned to new clusters based upon the updated cluster centroids. What distinguishes this version from the most common implementations is that this reassignment is repeated t_{max} times, regardless of whether the cluster centroids have converged or not. This choice was made so that a more fair comparison could be drawn with the other swarm based clustering algorithms which are limited by a predefined number of iterations.

B. Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) is a relatively recently discovered learning technique [2] that mimics the manner in which ants lay pheromone trails to inform the population of better solutions to the task at hand. This method of swarm-based intelligence may be used to create clusters whereby a population of agents learn from each other in which way to classify points. The algorithm implemented was found in [3].

Given a specified number of clusters n_c and a dataset D, an initial randomly generated clustering was generated, such that all clusters contained at least one point. The cluster centroids were then calculated as above in Eq 2. After centroids were calculated, an objective function was calculated for the clustering. For this implementation, it was simply the sum of squared distances between data points $\mathbf{x} \in C_i$ and their cluster centroids $\mathbf{c_i}$:

$$j_{hcm} = \sum_{i=1}^{n_c} \sum_{\mathbf{x} \in C_i} |\mathbf{c}_i - \mathbf{x}|^2.$$

If j_{hcm} was found to be smaller than any other j_{hcm} found, then a minimum objective function value j_{hcm_min} is updated as $j_{hcm\_min} = j_{hcm}$. (The initial value for j_{hcm_min} is ∞.) Initial "pheromones" were set to 1. They were then updated for each cluster i and datapoint k pair by

$$p_{i,k} = p_{i,k}(1-\rho) + \frac{u_{i,k}}{(j_{hcm} - j_{hcm\_min} + \epsilon)^\alpha}$$

where $u_{i,k} = 1$ if point k is in cluster i and $u_{i,k} = 0$ otherwise and ρ, ϵ, and α are all tunable parameters.

This process was repeated a specified number of times.

C. Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) seeks to mimic bird flocking or fish schooling behavior wherein a population of agents, referred to as particles, collectively inform one another about the optimality at varying places in the solution domain. The particular algorithm used for data clustering was taken from [4].

In this context a particle represents a set of cluster centroids. An initial population of particles was randomly generated across the solution domain using a uniform distribution. For each data point $\mathbf{x}$, the distance to each cluster centroid $c_{i,j}$ in each particle j was calculated:

$$d_{i,j}x = |\mathbf{x} - \mathbf{c}_{i,j}|.$$

The data points were then assigned to the cluster associated with the closest centroid and the objective function was calculated for each particle. In this context, this was the quantization error for each particle (see Eq 1 above).

Particles were then compared with previous performance (on a per-particle basis) and updated stochastically based upon a particle's position in the solution space that resulted in the lowest quantization error and upon the entire populations position of best performance via tunable parameters w, c_1, and $c2$.

D. DB-Scan

The DBScan algorithm was originally introduced by a group of researchers at the University of Munich to address several weaknesses of traditional clustering algorithms [5] and [6]. In particular, DBScan:

1) Does not require domain-specific knowledge
2) Can discover clusters of arbitrary shape
3) Is efficient on large data sets.

There are several variants of the DBScan algorithm, but all require two parameters: Eps, a radius value to determine whether points are in a given neighborhood, and $MinPts$, the minimum number of points necessary for a point to be considered a core point.

The first step was to categorize the points as being core, boundary, or noise points. A core point is a point q with a neighborhood function N_{Eps}, such that $|N_{Eps}(q)| \geq MinPts$. A boundary point is a point q with a neighborhood function N_{Eps}, such that $MinPts > |N_{Eps}(q)| \geq 1$. Lastly, a noise point is a point q with a neighborhood function N_{Eps}, such that $|N_{Eps}(q)| = MinPts$

Next, points were assigned to an arbitrary number of clusters. The algorithm iterated through each point $p \in Core$ and assigned an updated cluster label to p if a cluster label had not been assigned, and then assigned the same label to every p', where $p' \in N_{eps}(p)$.

That the number of clusters found by DBScan is unspecified is a major advantage of the algorithm, but it also means that values for Eps and $MinPts$ should be chosen carefully. Similarly, while DBScan is computationally efficient in that it doesn't require convergence or multiple iterations, the efficiency of the implementation of the neighborhood function can make a big difference in terms of performance. An additional consideration is that DBScan will produce different clusters depending on the order of the data, which may be an important consideration in some applications.

E. Competitive Learning

Competitive learning refers to a class of neural networks in which nodes "compete" to best fit a given set of inputs. Competitive learning was first introduced in 1985 by D. Rumelhart and D. Zipser [7].

The competitive learning network implemented here uses a topology with one input layer, one output layer and no hidden layers. Output layer weights were initialized using $U(0,1)$ draws, with the number of output nodes set to the number of unique classes in the original dataset, and all input vectors were normalized.

The training process involved randomly drawing input vectors from the data set and comparing these to the weights of each output node using a similarity function calculated as the inverse Euclidean distance between the two vectors. A bias consisting of the proportion of times that particular node had already "won" was added to minimize the problem of dead neurons, in which some output nodes are initialized with values such that they never "win." The output node with the

lowest adjusted similarity score was updated by adding the values of the input vector (scales by the tunable parameter η) to the weights of that node and then normalizing its weights. Convergence was considered to have occurred when the maximum difference between the node biases was less than a tolerance of 0.0005 for more than ten input vectors.

III. Experimental Approach

A. Datasets

A total of 10 seperate datasets were obtained from the UCI Machine Learning Repository, all of which represented classification problems [8]. However, the class feature was removed from each data set as the problem at hand is an *unsupervised learning* task. The data sets chosen were

1) Banknote Authentication
2) Wine
3) Iris
4) Seed
5) Wilt
6) Red Wine Quality
7) White Wine Quality
8) Breast Tissue
9) E. Coli
10) Haberman

B. Tuning

For each dataset, the number of clusters was chosen, naturally, to be the number of classes from the classification feature that was removed. Also, in the particular implementation of k-means clustering the authors chose, there were no true parameters to tune. For the other algorithms, final tuning parameters are provided here:

1) ACO:

TABLE I
ACO Tuning Results

Dataset	ρ	ϵ	α
Bank	0.1	0.1	1
Wine	0.005	0.3	1
Iris	0.1	0.5	0.5
Seed	0.1	0.8	0.8
Wilt	0.1	0.1	0.8
Red Wine	0.001	0.2	0.5
White Wine	0.001	0.5	0.5
Breast Tissue	10^{-5}	0.005	0.5
Ecoli	0.01	0.1	1
Haberman	0.1	0.01	1

2) PSO: For particle swarm optimization, no tuning parameters were found to result in higher performance than others on any data sets so the default parameters were left as those provided in [4] for all data sets: $w = 0.72$, $c_1 = 1.49$, and $c_2 = 1.49$.

3) DBScan: $MinPts$ was determined by the bformula $max(\lceil n_{pts}/n_{clust} \rceil, 2)$, where n_{pts} is the total number of input vectors and n_{clust} is the number of clusters expected [5]. Eps was determined heuristically based on MinPts and other attributes of the particular data set.

TABLE II
DBScan Tuning Results

Dataset	Eps	$MinPts$
Bank	2.2992	2
Wine	4.9237	2
Iris	0.9979	6
Seed	0.8902	2
Wilt	66.0275	2
Red Wine	2.1412	24
White Wine	2.7717	47
Breast Tissue	166.5705	2
Ecoli	0.3286	4
Haberman	5.9708	9

4) Competitive Learning: For competitive learning, η, the scaling factor for updates, was set to 0.25. The number of output nodes in a given network was determined by the number of unique classes in the original data set.

IV. Results

For each algorithm and each data set, 30 simulations were carried out with each iterated algorithm (that is, excluding DBScan) running for 1000 iterations. The mean quantization error and the standard deviation of these errors is provided below for each algorithm, data set combination.

A. k-Means

TABLE III
k-Means Performance

Dataset	Mean QE	SD
Bank Note	5.319	0.003
Wine	98.682	2.431
Iris	0.652	0.017
Seed	1.497	0.002
Wilt	97.283	0.005
White Wine	15.035	0.275
Red Wine	10.203	0.369
Breast	1638.252	7.572
Ecoli	0.198	0.007
Haberman	8.609	0.475

B. ACO

TABLE IV
ACO PERFORMANCE

Dataset	Mean QE	SD
Bank Note	7.213	0.006
Wine	256.856	4.537
Iris	1.925	0.020
Seed	3.264	0.023
Wilt	145.110	0.041
White Wine	38.957	0.019
Red Wine	27.808	0.043
Breast	8173.211	1099.845
Ecoli	0.390	0.002
Haberman	11.618	0.0211

C. PSO

TABLE V
PSO PERFORMANCE

Dataset	Mean QE	SD
Bank	70.077	13.741
Wine	20441.724	3624.536
Iris	2.298	0.553
Seed	8.102	1.605
Wilt	103838.658	37886.772
White Wine	3272.352	714.968
Red Wine	567.237	150.978
Breast Tissue	40625918.878	21276804.120
Ecoli	0.205	0.036
Haberman	261.332	59.826

D. DBScan

All standard deviations are zero here because there is no distributional component to the error. Each simulation results in the same clustering and thus, the same quantization error:

TABLE VI
DB-SCAN PERFORMANCE

Dataset	Mean QE	SD
Bank Note	6.059	0.000
Wine	48.898	0.000
Iris	1.519	0.000
Seeds	5.990	0.000
Wilt	246.054	0.000
White	26.460	0.000
Red	14.974	0.000
Breast	1120.206	0.000
E. Coli	0.775	0.000
Haberman	7.906	0.000

E. Competitive Learning

TABLE VII
COMPETITIVE LEARNING PERFORMANCE

Dataset	Mean QE	SD
Bank Note	12.108	0.634
Wine	149.847	22.106
Iris	1.268	0.176
Seeds	3.700	0.539
Wilt	172.562	13.718
White	33.066	3.553
Red	17.850	3.081
Breast	6034.870	1366.565
E. Coli	0.479	0.049
Haberman	16.493	0.000

V. CONCLUSION

One broad conclusion is that the particular PSO implemented does not appear to result in a valuable clustering mechnism. All quantization errors generated were far above those from the other algorithms. Thus, between PSO and ACO, ACO seems to be the better classifier (at least in this context). Between all the clustering algorithms excluding PSO, none seems an extremely better than the others. As expected, k-means and DBScan outperformed ACO and the competitive learning clustering methods. This is likely because of there parsimony. It was difficult to properly tune the clustering methods and this is of course an important component of their performance. It should also be noted that the distribution of quantization errors for the competitive learner implementation were more dispersed than the others. Thus, one would expect the competitive learner to be a less reliable clustering method.

REFERENCES

[1] Steinhaus, H. (1957). "Sur la Division des Corps Matriels en Parties". Bull. Acad. Polon. Sci. (in French) 4 (12): 801804.

[2] M. Dorigo, Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italy, 1992.

[3] Runkler, Thomas A. (2005). Ant Colony Optimization of Clustering Models. International Journal of Intelligent Systems, 20(12), 1233-1251.

[4] Merwe V. D. and Engelbrecht, A. P. (2003). Data Clustering Using Particle Swarm Optimization. Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003), Canbella, Australia, pp. 215-220.

[5] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, Jrg Sander (1999). OPTICS: Ordering Points To Identify the Clustering Structure. ACM SIGMOD international conference on Management of data. ACM Press. pp. 4960.y with OPTICS. Journal of chemical information and computer sciences, 42(3), 500-507.

[6] Martin Ester, Hans-Peter Kriegel, Jrg Sander, Xiaowei Xu (1996). Evangelos Simoudis, Jiawei Han, Usama M. Fayyad, eds. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226231. ISBN 1-57735-004-9.

[7] Rumelhart, David E., and David Zipser. "Feature discovery by competitive learning." Cognitive science 9.1 (1985): 75-112.

[8] Lichman, M. (2013). *UCI Machine Learning Repository* [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[9] Fenton, B., Sherrill, J. (2015). Git repository: https://github.com/joncheryl/csci447/tree/master/project4.