

Fuzzy Ants as a Clustering Concept

Parag M. Kanade and Lawrence O. Hall
 Dept. of Computer Science & Engineering, ENB118
 University of South Florida, Tampa FL 33620
pkanade@csee.usf.edu, hall@csee.usf.edu

Abstract

We present a swarm intelligence approach to data clustering. Data is clustered without initial knowledge of the number of clusters. Ant based clustering is used to initially create raw clusters and then these clusters are refined using the Fuzzy C Means algorithm. Initially the ants move the individual objects to form heaps. The centroids of these heaps are taken as the initial cluster centers and the Fuzzy C Means algorithm is used to refine these clusters. In the second stage the objects obtained from the Fuzzy C Means algorithm are hardened according to the maximum membership criteria to form new heaps. These new heaps are then sometimes moved and merged by the ants. The final clusters formed are refined by using the Fuzzy C Means algorithm. Results from three small data sets show that the partitions produced are competitive with those obtained from FCM.

Index Terms: Clustering, Ant Based Clustering, Swarm Intelligence, fuzzy c-means

1. Introduction

The aim of clustering is to separate a set of data points into self-similar groups such that the points that belong to the same group are more similar than the points belonging to different groups. Each group is called a cluster.

The clustering algorithms developed using the principals of Swarm Intelligence emphasize distributedness, flexibility, and robustness. The algorithms are based on direct or indirect feedback with relatively simple agents [2].

Each individual ant is a behaviorally simple agent with a limited memory. Even with limited memory and stochastic behavior ants consistently manage to perform several complicated tasks. The ant based clustering algorithms are based on the cemetery organization and larval sorting of ants. In ant colonies the workers form piles of corpses to clean up their nests. This aggregation of corpses is due to the attraction between the dead items. Small clusters of items grow by attracting workers to deposit more items; this positive feedback leads to the

formation of larger and larger clusters. Brood sorting is widespread in ant colonies. Worker ants gather larvae according to their size, all larvae of the same size tend to be clustered together. An item is dropped by the ant if it is surrounded by items which are similar to the item it is carrying; an item is picked up by the ant when it perceives items in the neighborhood which are dissimilar from the item to be picked up [2].

Data may be clustered using an iterative version of the Fuzzy C means (FCM) algorithm, but the draw back of FCM algorithm is that it is very sensitive to cluster center initialization because the search is based on the hill climbing heuristic [4]. We use the cluster centers obtained from the ant based algorithm as the initial cluster centers. This ant search is less sensitive to initialization because it is more global.

In past research the k-means clustering algorithm has been used on the centers obtained from the ant based algorithm; here we study the effect of using the Fuzzy C means approach on the cluster centers obtained from the ant based algorithm. We also do what we believe is more rigorous experimentation fully disclosing all parameters. In [1,9] the k-means algorithm was used to refine the clusters found by the ants. In [3] the ant system and the k-means algorithm were used for document clustering.

The outline of the paper is as follows: Section 2 explains the algorithm used in this paper, Section 3 explains the data sets used and the experimental results. A summary and discussion is in Section 4.

2. Algorithm

The general outline of the ant based algorithm used in this study was proposed in [1,9]. Initially the objects are scattered randomly on a discrete 2D board. The board can be considered as a matrix of $m \times m$ cells. The matrix is toroidal which allows the ants to travel from one end to another easily. The size of the board is dependent on the number of objects. We have used a board of $m \times m$ such that $m^2 = 4n$ where n is the total number of objects to be clustered. Initially the ants are randomly scattered

throughout the board. There are $n/3$ ants, where n is the total number of objects to be clustered.

The ants cluster the objects to form heaps. A heap is defined as a collection of 2 or more objects. A heap is spatially located in a single cell. The following parameters are defined for a heap and are used to construct heuristics for the clustering algorithm.

Consider a heap H with n_H objects, then we define the following parameters:

- The maximum distance between two s -dimensional objects in the heap

$$D_{max}(H) = \max_{X_i, X_j \in H} D(X_i, X_j)$$

Where D is euclidean the distance between the objects

- The center of mass of all the objects in the heap

$$O_{center}(H) = \frac{1}{n_H} \sum_{O_i \in H} O_i$$

- The most dissimilar object in the heap $O_{dissim}(H)$. It is the object which is the farthest from the center of the heap.

- The mean distance between the objects of H and the center of the mass of the heap

$$D_{mean}(H) = \frac{1}{n_H} \sum_{O_i \in H} D(O_i, O_{center}(H))$$

The main ant based clustering algorithm is presented in Figure 1[1].

1. Randomly place the ants on the board. Randomly place objects on the board at most one per cell
2. Repeat
3. For each ant Do
 - 3.1 Move the ant
 - 3.2 If the ant does not carry any object then if there is an object in the 8 neighboring cells of the ant, the ant possibly picks up the object,
 - 3.3 Else the ant possibly drops a carried object, by looking at the 8 neighboring cells around it
4. Until stopping criteria

Figure 1. The basic ant based algorithm [1]

Initially the ants are scattered randomly on the 2D board. The ant moves on the board and possibly picks an object or drops an object. The movement of the ant is not completely random. Initially the ant picks a direction randomly then the ant continues in the same direction

with a probability $P_{direction}$, otherwise it generates a new random direction. On reaching the new location on the board the ant may possibly pick up an object or drop an object, if it is carrying one. The heuristics and the exact mechanism for picking up or dropping an object are explained below. The stopping criterion for the ants, here, is the upper limit on the number of times through the repeat loop.

2.1. Picking up an object

When the ant is not carrying any object, it looks for possible objects to pick up by looking at the eight neighboring cells around its current position. If an object or heap is found then the ant possibly picks up an object. The heuristic for picking up an object depends on the number of objects in the heap. Three cases are considered: only one object, a heap of two objects and a heap of more than two objects. If a single object is present then the ant has a fixed probability of picking it up. If there is a heap of two objects then with a probability $P_{destroy}$ the ant destroys the heap by picking a random object from the heap. In the third case the ant picks up the most dissimilar object from the heap if the dissimilarity is above a given threshold T_{remove} .

The algorithm for picking up an object is given in Figure 2 [1].

1. Mark the 8 neighboring cells around the ant as “unexplored”
2. Repeat
 - 2.1 Consider the next unexplored cell around the ant
 - 2.2 If the cell is not empty then
 - 2.2.1 If the cell contains a single object X , then the object X is picked up with a probability P_{load} , else
 - 2.2.2 If the cell contains a heap of two objects, then the heap is destroyed by picking up a random object with a probability $P_{destroy}$ else
 - 2.2.3 If the cell contains a heap H of more than 2 objects, then the most dissimilar object, $O_{dissim}(H)$, of H is removed only if

$$\frac{D(O_{dissim}(H), O_{center}(H))}{D_{mean}(H)} > T_{remove}$$
 - 2.3 Label the cell as “explored”
3. Until all the neighboring cells have been explored or one object has been picked

Figure 2. Algorithm to pick up an object [1]

2.2. Dropping an object

When the ant is carrying an object, then it examines the 8 cells surrounding its current location. Three cases

are considered: the cell is empty, the cell contains one object only, and the cell contains a heap. In the first case the ant has a constant probability of dropping the object. In the second case a heap is created if the carried object is sufficiently similar to the one already in the cell. In the third case the ant will add its object to the heap if the object is closer to H 's center than the most dissimilar object of H . The algorithm for dropping the object is given in Figure 3 [1].

1. Mark the 8 neighboring cells around the ant as “unexplored”
2. Repeat
 - 2.1 Consider the next unexplored cell around the ant
 - 2.1.1 If the cell is empty then the object carried by the ant, X , is dropped with a probability P_{drop} else
 - 2.1.2 If the cell contains a single object X' then a heap of two objects is created by dropping X on X' only if $\frac{D(X, X')}{D_{max}} < T_{create}$ else
 - 2.1.3 If the cell contains a heap H then X is dropped on H only if $D(X, O_{center}(H)) < D(O_{dissim}(H), O_{center}(H))$
 - 2.2 Label c as “explored”
3. Until all the neighboring cells have been explored or the carried object has been dropped.

Figure 3. Algorithm for dropping an object [1]

2.3. The Second Stage

The ant based algorithm provides a relevant partition of data without any knowledge of the initial cluster centers. In the ant based algorithm if an object is a poor fit to a heap then it can take a long amount of time for it to be transported to a better heap/cluster. So in the past researchers have used ant based algorithms which are based on stochastic principles coupled with the k-means algorithm based on deterministic principles. We use the Fuzzy C means algorithm as the deterministic algorithm. The fuzzy C means algorithm requires good initializations, which are provided by the ant based algorithm.

One problem with the ant based algorithm not fixed by the FCM algorithm is that the number of classes is always overestimated. Many small homogenous heaps are formed. We use these heaps as the building blocks to build large heaps.

In the second stage we consider the heaps formed by the first stage and move the entire heap on the 2D board. The ants carry an entire heap of objects. The algorithm for picking up a heap is the same as that for the objects. Ants

will pick the heap with the same probability P_{load} . Ants drop a heap H_1 onto another heap H_2 provided that:

$$\frac{D(O_{center}(H_1), O_{center}(H_2))}{D_{max}} < T_{createforheap}$$

When two heaps H_1 and H_2 are merged, they form a single heap H_3 . They cannot be separated. The number of heaps goes on decreasing as the number of iterations increase.

The Fuzzy C Means algorithm is then used to cluster the data using the cluster centers obtained from the second stage of the ant based algorithm as an initialization. The algorithm used in the study is given in Figure 5.

1. Scatter the objects randomly on the board
2. Initialize the ants with random position, and random direction
3. For 1000 iterations Do
 - 3.1 For each ant Do
 - 3.1.1 Move the ant
 - 3.1.2 If the ant is carrying an object X then possibly drop the object X else
 - 3.1.3 Possibly pick up an object X
4. Use the cluster centers obtained in step 3 to initialize cluster centers for the Fuzzy C Means algorithm
5. Cluster the data using the Fuzzy C Means algorithm
6. Harden the data obtained from the Fuzzy C means algorithm, using the maximum membership criterion, to form new heaps
7. Repeat steps 1-6 by considering each heap as a single object

Figure 5. The 2 stage algorithm.

3. Experimental Results

The algorithm was applied to three small data sets the Iris Plant database, Wine Recognition database and Glass Identification database.

3.1. Iris plant database

This data set contains information about different types of Iris flowers. The data set consists of 150 examples each with four numeric predictive attributes. The data set contains 3 classes of 50 examples each, where each class refers to a type of Iris plant. One class is linearly separable from the other 2; the others have some overlap.

3.2. Wine recognition data

The data is the result of a chemical analysis of wines grown in a region in Italy but derived from three different cultivars. The dataset consists of 178 examples each with 13 continuous attributes. The data set contains 3 classes

with the following class distribution class 1: 59 examples class 2: 71 examples class 3: 48 examples.

3.3. Glass Identification Database

The data set consists of 214 examples each with 9 continuous attributes. There are 6 classes of glasses.

3.4. Experiments

The reported results are averaged for 50 runs of the experiments. In each run, the ants and the objects were initially placed at different positions on the board. The movement of ants and the picking and dropping of the objects also had a stochastic component. We performed experiments using three sets of parameters. By varying the parameters we controlled the number of heaps obtained. In the three experiments all but two parameters are fixed. The parameter is explored with multiple values for each dataset. The parameters and their ranges are shown in Table 1 except for $T_{createforheap}$ which is listed separately for each experiment. The results for the three sets of parameters for the datasets are shown below. The average results for 50 runs of the FCM algorithm with random initializations are in Table 2.

Table 1. Values of the parameters used in the experiments

Parameter	Range
Number of Iterations	1000
T_{create}	0.5
P_{drop}	0.2
$P_{destroy}$	0.3
P_{load}	0.3
T_{remove} (Iris)	1.5
T_{remove} (Wine)	3.0
T_{remove} (Glass)	2.0

Table 2. Results from the FCM algorithm

Data Set	Classes	Errors
Iris	3	16
Wine	3	56
Glass	6	84

Results for the Iris Data set are shown in Table 3. As $T_{createforheap}$ is increased fewer clusters are found. Every time two clusters are found, which is 17 times with the value of 0.14 and 42 times of the value of 0.18, at least 50 errors are observed. This is because the separable class is

almost always correctly grouped into one cluster with the others classes grouped together in a second cluster.

In the Iris feature space one class is linearly separable from the other two, so strictly speaking one could come to the determination that there are only two clusters. Hence, the results for the Iris data set considering 2 classes are shown in Table 4. In this case if you allow up to four classes to be found, while over clustering is done, you always get homogeneous clusters.

Table 3. Results for the Iris Database for three different parameter settings

$T_{createforheap}$	Classes Found	Errors before the second FCM	Errors after the second FCM
0.18	2.16	50.02	44.56
0.14	3	47.26	29
0.12	4.04	38.1	17.7

Table 4. Results for the Iris Data base considering 2 clusters, for three different parameter settings

$T_{createforheap}$	Classes Found	Errors before the second FCM	Errors after the second FCM
0.2	2.1	5.22	2.7
0.18	2.18	1.32	2.46
0.12	4.04	0	0

Results for the wine database are shown in Table 5.

Table 5. Results for the Wine Database for three different parameter settings

$T_{createforheap}$	Classes Found	Errors before the second FCM	Errors after the second FCM
0.075	2.2	65.82	61.64
0.085	3.6	58.1	54.48
0.1	4.64	56.5	51.96

Results for the glass database are shown in Table 6. The glass database has two broad categories of glass: window glass and non window glass. If we consider just these two classes then we get better results. FCM with random initializations has 20 errors in this case. The

results for three experiments with different parameter settings are shown in Table 7.

Table 6. Results for the Glass Database for three different parameter settings

$T_{createforheap}$	Classes Found	Errors before the second FCM	Errors after the second FCM
0.08	2.7	136.56	110.66
0.12	4.78	116.76	94.62
0.16	7.78	103.8	82.36

Table 7. Results for the Glass Database, considering two classes, for three different parameter settings

$T_{createforheap}$	Classes Found	Errors before the second FCM	Errors after the second FCM
0.12	4.78	25.06	19.28
0.16	2.7	49.58	21.3
0.18	2.18	49.9	27.22

4. Summary and Discussion

The use of ants in the clustering process is one way to determine the number of clusters. However, the ants are clearly sensitive to the threshold for deciding when to merge heaps and remove items from a heap. The original work in this area provided values, but no justification or way to set them. We have explored a range of values (albeit for one parameter) and shown how the results differ. We have not yet explored a systematic way to set the values of the parameters. The final partition found with 3 classes for the Iris data is always equivalent to what you get with FCM. However, the average value shows that sometimes you're getting two classes and sometimes you're getting four classes.

Essentially, the ants are finding the number of clusters and an initialization for FCM. They are not really producing a final partition. A difficulty with taking what they produce as a final partition is that nothing can be removed from a heap when heaps are being combined. This can be problematic.

For comparison purposes, Table 8 shows the number of times a partition of each class size was found for particular setting of parameters for the wine data set. It can be seen that the second application of FCM usually, but not always improves the partition. In this case, 3 classes are most often found.

In [1] the same three data sets we used make up a subset of those experimented with. They indicate that all parameters were randomly chosen from within set ranges for each of 50 runs. One of the ranges was for T_{remove} which was indicated to lie between 0.1 and 0.2.

Table 8. Typical example of variance in data partitions obtained with the wine data set with $T_{createforheaps} = 0.085$

Classes Found	Frequency	Errors Before the second FCM	Errors after the second FCM
2	7	57.57	61
3	19	58.79	56
4	14	58.29	53
5	8	57.5	49
6	1	54	49
7	1	55	50

However, this value must be greater than 1 since the furthest outlier will always be further from the centroid than the average distance of a member of the heap from the centroid. Also, we found that the clustering was highly sensitive to the values we call T_{create} and $T_{createforheap}$. These clearly have a strong influence on how many final clusters are obtained.

They indicate that the Iris Data had 178 examples which we believe is a typo because it is the same as the number of examples as in wine. However, with random choices of values they find an average 3.02 classes. They report an average of 23.1 errors which means they got very few two class partitions. Their reported numbers of errors are less than ours, but we believe that is because we more often get the higher error count two class partition.

They find an average of 3.06 classes on the wine data with an average of 9.57 errors. They find an average of 7.7 classes on the glass data with an average number of errors being 9.58. We do not believe such partitions are available with default k-means. In clustering into 3 classes utilizing the Wine data set, we find 53.65 errors over 50 random initializations. For the glass data set, with 7 classes we find 92.5 errors on average over 50 random initializations and with 8 classes we find 90.2 errors on average. We cannot get close to such low error partitions with initializations of FCM, either. Later, they show results from the k-means algorithm starting with 10 classes. They appear to be taking "degenerate" partitions because they quote an average number of classes of 6.954 Iris, 8.98 for wine and 7.06 for glass respectively. This strikes us as unusual.

In [3], a simpler approach than the one we discuss is used to cluster documents. The authors utilize k-means after the ants find the initial cluster centers and find the

number of centers. It was startling to us that, over 10 runs, the ants find on average a non-fractional number of clusters. Utilizing the ant based initialization, the accuracy of the final cluster partition was better than just using k-means. They indicated that their implementation of k-means was sensitive to the order of data, which suggests a nonstandard implementation.

Some early work on ant sorting which can be applied to exploratory data analysis or clustering was done in [5] and [6]. The results were promising and built on by [3].

In [7] an ant colony optimization approach is adapted for clustering and favorably compared with a genetic approach to clustering and k-means. It looks at the problem as a version of the traveling salesman problem and is hence very different from our approach. In [8] a somewhat simpler algorithmic approach to ant based clustering is applied to document clustering. They stress that their approach is closely aligned with actual ant behavior. They show good results in an artificial domain and for one document clustering example (using a sophisticated representation for documents). Their results are similar to those in [6], but their algorithm is simpler.

Future work should focus on setting parameters automatically. Further work should be done on whether this approach can reliably produce the "right" number of clusters. It is possible that modifications to allow ants to remove elements from heaps in the final stage would allow a reasonable partition without a conventional clustering algorithm. One might fuzzify the thresholds.

Another avenue we are pursuing is to allow the ants to relocate cluster centroids in features space. The formulation is the same at a very high-level as was done in [4], but ants are utilized rather than a genetic approach.

Acknowledgments: This research partially supported by The National Institutes of Health via a bioengineering research partnership under grant number 1 R01 EB00822-01.

References

- [1] Monmarche N., Silmane M., and Venturini G., "AntClass: discovery of clusters in numeric data by an hybridization of an ant colony with k-means algorithm", *Internal Report no. 213*, Laboratoire d'Informatique de l'Universite de January 1999.
- [2] Bonabeau E., Dorigo M., and Theraulaz G., *Swarm Intelligence: From natural to artificial systems*, Oxford University press, NY, 1999
- [3] Bin W., Shaohui L., and Zhongzhi S., "CSIM: A document clustering algorithm based on swarm intelligence", *Evolutionary Computation*, 2002. CEC '02. Proceedings of the 2002 Congress on, Volume: 1, 12-17 May 2002 Page(s):477-482
- [4] Hall, L.O., Ozyurt, I.B., and Bezdek, J.C., "Clustering with a Genetically Optimized Approach", *IEEE Transactions on Evolutionary Computation*, V. 3, No. 2, pp. 103-112, 1999.

[5] J.L. Deneubourg, S. Goss, N. Francs, A. Sendova-Franks, C. Detrain and L. Chretien. "The dynamics of collective sorting: Robot-Like Ant and Ant-Like Robot." In *Proceedings First Conference on Simulation of adaptive Behavior: from animals to animats*, edited by J.A. Meyer and S.W. Wilson, pp. 356-365. Cambridge, MA: MIT press, 1991.

[6] E. Lumer and B. Faieta, "Diversity and Adaptation in Populations of Clustering Ants." In *Proceedings Third International Conference on Simulation of Adaptive Behavior: from animals to animats 3*, 499-508. Cambridge, Massachusetts MIT press, 1994.

[7] C-F Tsai ; H-C Wu; C-W Tsai, "A new data clustering approach for data mining in large databases", *Proceedings International Symposium on Parallel Architectures, Algorithms and Networks. I-SPAN'02*, Pages 315-320, 2002.

[8] V. Ramos and J.J. Merelo, Self-Organized Stigmergic Document Maps: Environment as a Mechanism for Context Learning, in *AEB 2002 – 1st Spanish Conference on Evolutionary and Bio-Inspired Algorithms*, E. Alba, F. Herrera, J.J. Merelo et al. (Eds.), pp. 284-293, Centro Univ. de Mérida, Mérida, Spain, 6-8 Feb. 2002.

[9] N. Monmarche, M. Slimane, G. Venturini, "On improving clustering in numerical databases with artificial ants", *Advances in Artificial Life. 5th European Conference, ECAL99. Proceedings (Lecture Notes in Artificial Intelligence Vol. 1674)*, Pages 626-635 (1999).