6-17-2004

Fuzzy Ants as a Clustering Concept

Parag M. Kanade
University of South Florida

Follow this and additional works at: <https://digitalcommons.usf.edu/etd>

Part of the [American Studies Commons](#)

Scholar Commons Citation

Kanade, Parag M., "Fuzzy Ants as a Clustering Concept" (2004). *USF Tampa Graduate Theses and Dissertations*.

<https://digitalcommons.usf.edu/etd/1104>

This Thesis is brought to you for free and open access by the USF Graduate Theses and Dissertations at Digital Commons @ University of South Florida. It has been accepted for inclusion in USF Tampa Graduate Theses and Dissertations by an authorized administrator of Digital Commons @ University of South Florida. For more information, please contact digitalcommons@usf.edu.

Fuzzy Ants as a Clustering Concept

by

Parag M. Kanade

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Lawrence O. Hall, Ph.D.
Dmitry B. Goldgof, Ph.D.
Sudeep Sarkar, Ph.D.

Date of Approval:
June 17, 2004

Keywords: Cluster Analysis, Swarm Intelligence, Ant Colony Optimization, Fuzzy C Means
Algorithm, Hard C Means Algorithm

© Copyright 2004, Parag M. Kanade

DEDICATION

To my parents and my loving sister Pallavi.

ACKNOWLEDGEMENTS

I would like to thank Dr. Lawrence O. Hall for giving me the opportunity to work with him. His guidance and support was invaluable. I would like to thank Dr. Dmitry Goldgof, and Dr. Sudeep Sarkar for taking the time and effort to ensure that my work was of quality. I am grateful to my family, friends and colleagues who gave me inspiration and helped me in every step.

TABLE OF CONTENTS

LIST OF TABLES	iii
LIST OF FIGURES	vi
ABSTRACT	viii
CHAPTER 1 INTRODUCTION	1
1.1 Clustering	1
1.1.1 Hard Clustering	2
1.1.2 Fuzzy Clustering	3
1.2 Swarm Intelligence	3
1.2.1 Self Organization in Social Insects	4
1.2.2 Stigmergy	5
1.2.3 Cemetery Organization and Brood Sorting in Ants	5
CHAPTER 2 DATASETS	7
2.1 Iris Plant Dataset	8
2.2 Wine Recognition Dataset	8
2.3 Glass Identification Dataset	13
2.4 Multiple Sclerosis Dataset	14
2.5 MRI and British Towns' Datasets	14
2.6 Artificial Datasets	14
CHAPTER 3 FUZZY ANTS ALGORITHM	15
3.1 Picking up an Object	20
3.2 Dropping an Object	20
3.3 The Second Stage	21
3.4 Results	23
3.4.1 Iris Dataset	23
3.4.2 Wine Recognition Dataset	24
3.4.3 Glass Identification Dataset	25
3.4.4 Artificial Datasets	26
3.5 Effect of Parameter Variation	29
3.6 Variations in the Algorithm	29
3.7 Discussion	36
CHAPTER 4 AUTOMATICALLY SETTING THE PARAMETERS	38
4.1 New Metric for Merging Heaps	38
4.2 Results Using the New Metric	38
4.3 Fuzzy Hypervolume	40

CHAPTER 5	FUZZY ANT CLUSTERING WITH CENTROIDS	44
5.1	Introduction	44
5.2	Reformulation of Clustering Criteria for FCM and HCM	44
5.3	Algorithm	45
5.4	Results	49
5.5	Hard C Means	51
5.6	Execution Time	53
5.7	Conclusions	55
CHAPTER 6	CONCLUSIONS AND FUTURE WORK	57
6.1	Combined Algorithm	57
6.2	Contributions	59
6.3	Future Work	59
REFERENCES		60
APPENDICES		63
Appendix A	Datasets	64
A.1	Parameters Used to Generate the Gaussian Datasets	64
A.2	Plots of the Different Datasets	66

LIST OF TABLES

Table 2.1	Datasets	7
Table 3.1	Values of the Parameters Used in the Fuzzy Ants Experiments	23
Table 3.2	Results from FCM	24
Table 3.3	Results for Iris Dataset	24
Table 3.4	Results for Iris Dataset (2 Class)	25
Table 3.5	Results for Wine Recognition Dataset	25
Table 3.6	Results for Glass Identification Dataset	25
Table 3.7	Results for Glass Identification Dataset (2 Class)	26
Table 3.8	Results for Gauss-1 Dataset	27
Table 3.9	Results for Gauss-2 Dataset	27
Table 3.10	Results for Gauss-3 Dataset	27
Table 3.11	Results for Gauss-4 Dataset	27
Table 3.12	Results for Gauss-5 Dataset	28
Table 3.13	Results for Gauss500-1 Dataset	28
Table 3.14	Results for Gauss500-2 Dataset	28
Table 3.15	Results for Gauss500-3 Dataset	28
Table 3.16	Results for Gauss500-4 Dataset	29
Table 3.17	Results for Gauss500-5 Dataset	29
Table 3.18	Effect of Variation of Iterations on Iris Dataset	30
Table 3.19	Effect of Variation of Iterations on Wine Dataset	30
Table 3.20	Results for the Iris Dataset (3D Board)	31
Table 3.21	Results for the Iris Dataset (2 Class) (3D board)	31
Table 3.22	Results for the Wine Dataset (3D Board)	32
Table 3.23	Results for the Glass Dataset (3D board)	32

Table 3.24	Results for the Glass (2 Class) Dataset (3D Board)	33
Table 3.25	Results for the Gauss500-1 Dataset (3D Board)	34
Table 3.26	Results for the Gauss500-2 Dataset (3D Board)	34
Table 3.27	Results for the Gauss500-3 Dataset (3D Board)	35
Table 3.28	Results for the Gauss500-4 Dataset (3D Board)	35
Table 3.29	Results for the Gauss500-5 Dataset (3D Board)	36
Table 3.30	Typical Example of Variance in Data Partitions Obtained with the Wine Dataset with $T_{createforheap} = 0.09$ and 3000 Iterations	37
Table 4.1	Results from the New Metric for Iris Dataset	39
Table 4.2	Results from the New Metric for Wine Dataset	39
Table 4.3	Effect of Variation of Iterations on Iris Dataset	40
Table 4.4	Effect of Variation of Iterations on Wine Dataset	40
Table 4.5	$T_{createforheap}$ and Fuzzy Hypervolume	42
Table 5.1	Parameter Values	49
Table 5.2	Results for Fuzzy C Means	50
Table 5.3	New Parameters	50
Table 5.4	Results for FCM Obtained from Modified Parameters	51
Table 5.5	Results for Hard C Means	52
Table 5.6	Results for the British Towns Dataset	52
Table 5.7	Results for the Wine Dataset	53
Table 5.8	Results for the MRI Dataset	53
Table 5.9	Variation of R_m with the Number of Ants for Slice # 35 of MRI Dataset	53
Table 5.10	Execution Time	54
Table 5.11	Frequency of Different Extrema from FCM, for British Towns and MRI Datasets	55
Table 5.12	Frequency of Different Extrema from HCM, for Glass (2 Class) and Iris Datasets	56
Table 5.13	Frequency of Different Extrema from HCM, MRI Dataset	56
Table 6.1	Results from the Combined Algorithm	58
Table 6.2	Results from the Combined Algorithm (Errors)	58
Table A.1	Parameters Used to Generate the Gauss Datasets	64

LIST OF FIGURES

Figure 1.1	Fuzzy C Means Algorithm	4
Figure 2.1	Iris Dataset (Normalized)- First 2 Principal Components	9
Figure 2.2	Iris Dataset (Normalized)- First 3 Principal Components	10
Figure 2.3	Iris Dataset: 2 Classes (Normalized)- First 2 Principal Components	11
Figure 2.4	Iris Dataset: 2 Classes (Normalized)- First 3 Principal Components	12
Figure 3.1	The Ant Based Algorithm	16
Figure 3.2	Algorithm	18
Figure 3.3	Algorithm (cont...)	19
Figure 3.4	Algorithm to Pick up an Object	20
Figure 3.5	Algorithm for Dropping an Object	21
Figure 3.6	The Two-stage Algorithm	22
Figure 4.1	$T_{createforheap}$ vs Fuzzy Hypervolume	42
Figure 4.2	$T_{createforheap}$ vs Fuzzy Hypervolume (Log)	43
Figure 5.1	Pictorial View of the Algorithm	47
Figure 5.2	Fuzzy Ant Clustering with Centroids Algorithm	48
Figure A.1	Wine Dataset (Normalized)- First 2 Principal Components	66
Figure A.2	Wine Dataset (Normalized)- First 3 Principal Components	66
Figure A.3	Glass Dataset (Normalized)- First 2 Principal Components	67
Figure A.4	Glass Dataset (Normalized)- First 3 Principal Components	67
Figure A.5	Glass Dataset: 2 classes (Normalized)- First 2 Principal Components	68
Figure A.6	Glass Dataset: 2 classes (Normalized)- First 3 Principal Components	68
Figure A.7	Multiple Sclerosis Dataset (Normalized)- First 2 Principal Components	69
Figure A.8	Multiple Sclerosis Dataset (Normalized)- First 3 Principal Components	69
Figure A.9	Gauss-1 Dataset (Normalized)	70

Figure A.10	Gauss-2 Dataset (Normalized)	70
Figure A.11	Gauss-3 Dataset (Normalized)	71
Figure A.12	Gauss-4 Dataset (Normalized)	71
Figure A.13	Gauss-5 Dataset (Normalized)	72
Figure A.14	Gauss500-1 Dataset (Normalized)	72
Figure A.15	Gauss500-2 Dataset (Normalized)	73
Figure A.16	Gauss500-3 Dataset (Normalized)	73
Figure A.17	Gauss500-4 Dataset (Normalized)	74
Figure A.18	Gauss500-5 Dataset (Normalized)	74

FUZZY ANTS AS A CLUSTERING CONCEPT

Parag M. Kanade

ABSTRACT

We present two Swarm Intelligence based approaches for data clustering. The first algorithm, *Fuzzy Ants*, presented in this thesis clusters data without the initial knowledge of the number of clusters. It is a two stage algorithm. In the first stage the ants cluster data to initially create raw clusters which are refined using the Fuzzy C Means algorithm. Initially, the ants move the individual objects to form heaps. The centroids of these heaps are refined by the Fuzzy C Means algorithm. 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 moved by the ants. The final clusters formed are refined by using the Fuzzy C Means algorithm. Results from experiments with 13 datasets show that the partitions produced are competitive with those from FCM. The second algorithm, *Fuzzy ant clustering with centroids*, is also a two stage algorithm, it requires an initial knowledge of the number of clusters in the data. In the first stage of the algorithm ants move the cluster centers in feature space. The cluster centers found by the ants are evaluated using a reformulated Fuzzy C Means criterion. In the second stage the best cluster centers found are used as the initial cluster centers for the Fuzzy C Means algorithm. Results on 18 datasets show that the partitions found by FCM using the ant initialization are better than those from randomly initialized FCM. Hard C Means was also used in the second stage and the partitions from the ant algorithm are better than from randomly initialized Hard C Means. The *Fuzzy Ants* algorithm is a novel method to find the number of clusters in the data and also provides good initializations for the FCM and HCM algorithms. We performed sensitivity analysis on the controlling parameters and found the *Fuzzy Ants* algorithm to be very sensitive to the $T_{createforheap}$ parameter. The FCM and HCM algorithms, with random initializations can get stuck in a *bad* extrema, the *Fuzzy ant clustering with centroids* algorithm successfully avoids these *bad* extremas.

CHAPTER 1

INTRODUCTION

Modern technology provides us with efficient and low-cost techniques for data collection. Raw data, however, is of limited use for decision making and intelligent analysis. Machine learning aims to create automatic or semiautomatic tools for the analysis of raw data to discover useful patterns and rules. Clustering is one of the most important unsupervised learning techniques.

1.1 Clustering

The aim of cluster analysis is to find groupings or structures within unlabeled data [19]. The partitions found should in general have the following properties

Homogeneity: The data that are assigned to the same cluster should be similar.

Heterogeneity: The data that are assigned to different clusters should be different.

In most cases the data is in the form of real-valued vectors. The euclidean distance is a suitable measure of similarity for these datasets. The partitions should then be such that the intra-cluster distance is minimized and the inter-cluster distance is maximized.

The clustering techniques can be broadly classified as follows [16]:

- *Incomplete/Heuristic:* Clusters are determined by heuristic methods based on visualization after dimensionality reduction of the data. The dimensionality is reduced by using geometrical methods (PCA) or projection techniques.
- *Deterministic crisp:* Each datum is assigned to one and only one cluster.
- *Overlapping crisp:* Each datum can be simultaneously assigned to several clusters.
- *Probabilistic:* A probability distribution specifies the probability of assignment of the datum to a cluster.

- *Possibilistic/Fuzzy*: These are pure fuzzy clustering techniques. Degrees of membership indicate the extent to which the datum belongs to the cluster. The sum of memberships of each datum across all the clusters may not be 1 for possibilistic clustering.
- *Hierarchical*: These techniques are based on either dividing the data into more fine-grained classes or combining small classes to more coarse-grained classes.
- *Objective function based*: These techniques are based on an objective or evaluation function that assigns a quality or error value to each possible partition or group of clusters. The partition that obtains the best evaluation is the chosen solution.
- *Cluster estimation*: These techniques use heuristic functions to build partitions and estimate the cluster parameters.

1.1.1 Hard Clustering

Hard C Means (HCM) is one of the simplest unsupervised clustering algorithms to cluster data into a fixed number of clusters. The basic idea of the algorithm is to initially guess the centroids of the clusters and then refine them. Cluster initialization is very crucial because the algorithm is very sensitive to this initialization. A good choice for the initial cluster centers is to place them as far away from each other as possible. The nearest neighbor algorithm is then used to assign each feature vector to a cluster. Using the clusters obtained, new cluster centroids are calculated. The above steps are repeated until there is no significant change in the centroids.

The objective function minimized by the hard C Means algorithm is given in 1.1

$$J = \sum_{i=1}^c \sum_{k=1}^n D_{ik}(x_k, \beta_i) \quad (1.1)$$

where

- $c \geq 2$: Number of clusters
- n : Number of data points
- β_i : The i^{th} cluster prototype
- $D_{ik}(x_k, \beta_i)$: Distance of x_k from i^{th} cluster center

1.1.2 Fuzzy Clustering

Hard clustering algorithms assign each datum to one and only one cluster. This model is inappropriate for real datasets in which the boundaries between the clusters may be fuzzy. Fuzzy algorithms can assign data to multiple clusters. Fuzzy algorithms are based on fuzzy logic which was developed by Lotfi Zadeh [38]. Deterministic membership functions assign each datum to a particular cluster, that is, the membership functions map the membership of the datum in a cluster to either 0 or 1. Fuzzy membership functions, on the other hand, map the memberships to the real interval $[0.0 \ 1.0]$. The degree of membership in the cluster depends on the closeness of the datum to the cluster center. High membership values indicate less distance between the datum and the cluster center.

The Fuzzy C Means algorithm (FCM), developed by Bezdek [3], allows the datum to be a partial member of more than one cluster. The FCM algorithm is based on minimizing the objective function 1.2

$$J_m(U, \beta) = \sum_{i=1}^c \sum_{k=1}^n U_{ik}^m D_{ik}(x_k, \beta_i) \quad (1.2)$$

where

- $c \geq 2$: Number of clusters
- n : Number of data points
- β_i : The i^{th} cluster prototype
- $D_{ik}(x_k, \beta_i)$: Distance of x_k from i^{th} cluster center
- U_{ik} : Membership of the k^{th} object in the i^{th} cluster
- $m \geq 1$: The degree of fuzzification

The Fuzzy C Means algorithm is shown in Figure 1.1

The drawback of clustering algorithms like FCM and HCM, which are based on the hill climbing heuristic is, prior knowledge of the number of clusters in the data is required and they have significant sensitivity to cluster center initialization.

1.2 Swarm Intelligence

Research in using the social insect metaphor for solving problems is still in its infancy. The systems developed using swarm intelligence principles emphasize distributiveness, direct or indirect

Figure 1.1. Fuzzy C Means Algorithm

interactions among relatively simple agents, flexibility and robustness [7]. Successful applications have been developed in the communication networks, robotics and combinatorial optimization fields.

1.2.1 Self Organization in Social Insects

Self Organization is the mechanism of forming global structures from the interaction of lower-level components. The rules specifying the interactions among lower-level components are based on local information, without reference to global information. The global pattern is an emergent property of the system. Self organization is based on four basic principles [7]:

- *Positive feedback*: It is the basis of morphogenesis. Examples of positive feedback are recruitment and reinforcement.
- *Negative feedback*: It counterbalances positive feedback and is responsible for the stabilization of the emergent pattern. Negative feedback may be in the form of saturation, exhaustion, or competition.
- *Amplification of fluctuations*: Amplification of fluctuations is an important factor for self organization. Randomness is crucial, since it enables discovery of new solutions. Fluctuations can act as seeds for the growth of new structures.
- *Multiple interactions*: Self organization is a result of interactions within mutually tolerant individuals. Individuals make use of their own activities as well as of others' activities. For

examples the trails laid by ants appear to self-organize and be used collectively if individuals use others' pheromone.

The key characteristics of a self organized system are [7]:

- *Structures*: Self organized systems create spatiotemporal structures in an initially homogeneous medium. Nest architectures, foraging trails, and social organization are the examples of spatiotemporal structures.
- *Coexistence*: Coexistence of several stable states is possible. The emergent system is the result of amplification of random deviations, depending on the initial conditions the system converges to one among several possible stable states.
- *Bifurcations*: Depending on the variation in the parameters, the system may bifurcate. The behavior of the system changes drastically at bifurcations.

1.2.2 Stigmergy

Self-organization in social insects is the result of direct and indirect interactions. Examples of direct interactions are visual contact, chemical contact, antennation, trophallaxis, and mandibular contact [7]. Indirect interactions occur when one individual changes the environment and the others respond to the new environment at a later time. This subtle interaction is an example of stigmergy.

Stigmergy helps in replacing coordination through direct communications by indirect interactions, reducing the communication among agents. Stigmergy also allows the emergent system to be flexible. The insects respond to an external perturbation as if it were a modification of the environment caused by the colony's activities. That is, the colony collectively responds to the perturbation [7]. Artificial agents designed using this principle can respond to a perturbation without being reprogrammed to deal with that particular perturbation.

1.2.3 Cemetery Organization and Brood Sorting in Ants

Many species of ants cluster dead bodies to form cemeteries, and sort the larvae into several piles. This behavior can be simulated using a simple model in which the agents move randomly in space and pick up and deposit items on the basis of local information. The clustering and sorting behavior of ants can be used as a metaphor for designing new algorithms for data analysis and graph partitioning. The objects can be considered as items to be sorted. Objects placed next to

each other have similar attributes. This sorting takes place in two-dimensional space, offering a low-dimensional representation of the objects.

The cemetery organization of the ant *Lasius niger* and *Pheidole pallidula* has been well studied [7]. The ants form piles of corpses, cemeteries, to clean up their nests. It is observed that if corpses are randomly distributed, the workers form cemetery clusters. If the area is not large or if there are spatial heterogeneities, the clusters are formed along the edges or following the heterogeneities [7]. This aggregation phenomenon is due to the attraction between the dead items mediated by the worker ants. 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 also widespread in ants. It is observed in the ant *Leptothorax unifasciatus* [7]. Worker ants gather larvae according to their size, all larvae of the same size tend to cluster together. An item is dropped by the ant if it is surrounded by items similar to the item the ant is carrying, an item is picked up by the ant when it perceives items in the neighborhood which are dissimilar to the items to be picked up. In *Leptothorax unifasciatus* the small larvae are located in the center and larger larvae are located toward the periphery. The amount of space allocated to a brood item varies with the type of brood. Eggs and micro-larvae are clustered compactly at the center, with each individual item given little individual space. Individual space tends to increase toward the periphery. The largest larvae, which are located at the periphery, are allocated more individual space [7].

CHAPTER 2

DATASETS

Six real datasets and ten artificial datasets were tested in this thesis. The datasets tested were

- *Iris Plant Dataset*
- *Wine Recognition Dataset*
- *Glass Identification Dataset*
- *Multiple Sclerosis Dataset*
- *MRI Dataset*
- *British Towns' Dataset*
- *Gauss 1-5 Datasets*
- *Gauss500 1-5 Datasets*

The datasets are described in Table 2.1 and the following sections.

Table 2.1. Datasets

Dataset	Number of Examples	Number of Continuous Attributes	Number of Classes
Iris	150	4	3
Wine	178	13	3
Glass	214	9	6
MRI	65536	3	3
Multiple Sclerosis	98	5	2
British Towns	50	4	5
Gauss1-5	1000	2	5
Gauss500-1-5	500	2	5

2.1 Iris Plant Dataset

This is probably the most studied dataset in the field of pattern recognition. The dataset has information about different types of Iris flowers [6].

The dataset contains 3 classes, *Iris Setosa*, *Iris Versicolour*, and *Iris Virginica*, with 50 instances of each class. One class is linearly separable from the other two. The remaining classes are not linearly separable from each other. There are four numeric predictive attributes for each instance, they are

- *Petal Length*
- *Petal Width*
- *Sepal Length*
- *Sepal Width*

To visualize the dataset, the Principal Component Analysis (PCA) algorithm [11] is used to project the data points to a 2D and 3D space. Figures 2.1 and 2.2 show the scatter of the points, after PCA, in 2D and 3D respectively. One class is linearly separable from the other two. For clustering purposes, the iris dataset can be considered having only 2 clusters. The dataset with 2 clusters is shown in Figures 2.3 and 2.4.

2.2 Wine Recognition Dataset

The data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. Quantities of 13 different constituents found in each of the three types of wines was measured [6]. The constituents are:

- *Alcohol*
- *Malic acid*
- *Ash*
- *Alcalinity of ash*
- *Magnesium*
- *Total phenols*

Figure 2.1. Iris Dataset (Normalized)- First 2 Principal Components

Note: All values are in generic units

Figure 2.2. Iris Dataset (Normalized)- First 3 Principal Components

Note: All values are in generic units

Figure 2.3. Iris Dataset: 2 Classes (Normalized)- First 2 Principal Components

Note: All values are in generic units

Figure 2.4. Iris Dataset: 2 Classes (Normalized)- First 3 Principal Components

Note: All values are in generic units

- *Flavanoids*
- *Nonflavanoid phenols*
- *Proanthocyanins*
- *Color intensity*
- *Hue*
- *OD280/OD315 of diluted wines*
- *Proline*

The class distribution is *class 1*: 59 *class 2*: 71 and *class 3*: 48. The projection of the data points in 2D and 3D space is shown in appendix A, in Figures A.1 and A.2 respectively.

2.3 Glass Identification Dataset

The study of types of glass was motivated by its immense potential in criminology. The glass at the scene of crime, if identified correctly, can be used as evidence. The dataset has 214 examples of six different types of glass. Each example has 9 attributes, giving the quantity of different chemical elements present in the glass [6]. The attributes are:

- *Refractive Index*
- *Na: Sodium*
- *Mg: Magnesium*
- *Al: Aluminum*
- *Si: Silicon*
- *K: Potassium*
- *Ca: Calcium*
- *Ba: Barium*
- *Fe: Iron*

The projection of the data points in 2D and 3D space is shown in Figures A.3 and A.4 respectively. The glass dataset can be considered as having two classes *window glass* and *non-window glass*. The projection of the data considering two classes is shown in Figures A.5 and A.6.

2.4 Multiple Sclerosis Dataset

The Multiple Sclerosis dataset has 98 instances each having 5 numeric continuous attributes. The projection of the data points in 2D and 3D space is shown in Figures A.7 and A.8 respectively.

2.5 MRI and British Towns' Datasets

The MRI dataset has 65,536 instances, each with 3 numeric continuous attributes, from 3 classes. The British Towns' dataset has 50 instances, each with 4 numeric continuous attributes, from 5 classes.

2.6 Artificial Datasets

Ten artificial datasets were generated using Gaussian distributions. A mixture of five Gaussians was used to generate the data. The probability distribution across all the datasets is the same but the means and standard deviations of the Gaussians are different. Of the ten datasets, five datasets had 500 instances each and the remaining five datasets had 1000 instances each. Each example had two attributes. The datasets are shown in Figures A.9–A.18. The parameters used to generate the datasets are shown in the Appendix A.1.

CHAPTER 3

FUZZY ANTS ALGORITHM

The *Fuzzy Ants* algorithm is based on *Swarm Intelligence* principles applied to cluster data. Data is clustered without the initial knowledge of the number of clusters. We use ant based clustering 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 criterion 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.

In past research the K means clustering algorithm has been used on the centers obtained from the ant based algorithm as introduced in [26]; here we study the effect of using the Fuzzy C means approach on the cluster centers obtained from the ant based algorithm. In [26] the Fuzzy K-means algorithm was used to refine the clusters found by the ants. In [4] the ant system and the K-means algorithm were used for document clustering.

The general outline of the ant based algorithm used in this study was proposed in [26]. Initially the objects are scattered randomly on a discrete 2D board. The board can be considered 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 $\frac{n}{3}$ 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.

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 the euclidean 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 3.1 [26].

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 3.1. The Ant Based Algorithm

Initially the ants are scattered randomly on the 2D board. The ant moves on the board and possibly picks up 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 number of times through the repeat loop.

The different steps of the algorithm are shown in Figures 3.2 and 3.3.

2D Board

Initial Stage

Figure 3.2. Algorithm

1st Stage

2nd Stage

Figure 3.3. Algorithm (cont...)

3.1 Picking up an Object

When the ant is not carrying any object, it searches for possible objects to pick up by examining 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 3.4 [26].

Figure 3.4. Algorithm to Pick up an Object

3.2 Dropping an Object

When the ant is carrying an object, 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 object carried 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 3.5 [26].

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 3.5. Algorithm for Dropping an Object

3.3 The Second Stage

The ant based algorithm provides a partition of the 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 in combination with the K-means algorithm which is based on deterministic principles. We have used the Fuzzy C means algorithm as the deterministic algorithm. The fuzzy C means algorithm requires good initializations, which can be 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 homogeneous 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 clustered together, they form only one heap H_3 . They cannot be separated any more. The number of heaps either decrease or remain constant 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 3.6.

1. Scatter the objects randomly on the board
2. Initialize the ants with random position, and random direction
3. For $N_{iterations}$ 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 3.6. The Two-stage Algorithm

The values of the various parameters as used here are shown in Table 3.1.

Table 3.1. Values of the Parameters Used in the Experiments

Parameter	Value
$N_{iterations}$	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
T_{remove} (Gauss)	1.5

3.4 Results

The reported results are averaged for 50 runs of the experiments. In each run, the ants and the objects were initially randomly 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 different parameter values. By varying the parameters we could control the number of heaps obtained. In the experiments all but one parameter is fixed. The parameter $T_{createforheap}$ is varied. This parameter is the threshold for the maximum dissimilarity allowed while merging the heaps. The original parameters are shown in Table 3.1, the results for different parameter values are shown below. The average results for 50 runs of the FCM algorithm with random initializations are in Table 3.2. We report errors which may not be the fairest measure given the algorithm is optimizing a function which indirectly relates to errors.

3.4.1 Iris Dataset

Results for the Iris Data set are shown in Table 3.3. As $T_{createforheap}$ is increased fewer clusters are found. Every time two clusters are found, which is 3 times with the value of 0.12, 19 times with the value of 0.14, and 41 times with the value of 0.18, at least 50 errors are observed. This is because the separable class is almost always correctly split off with the other cluster consisting of a mixture of the other two clusters.

In the Iris dataset 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

Table 3.2. Results for FCM Algorithm (Avg. from 50 random initializations)

Dataset	Classes	Errors
Iris	3	16
Iris	2	3
Wine	3	9
Glass	6	96
Glass	2	20
Gauss 1	5	0
Gauss 2	5	0
Gauss 3	5	0
Gauss 4	5	25.32
Gauss 5	5	7
Gauss 500-1	5	1
Gauss 500-2	5	0
Gauss 500-3	5	1
Gauss 500-4	5	1.94
Gauss 500-5	5	0

for the Iris dataset considering 2 classes are shown in Table 3.4. In this case if we allow up to four classes to be found, while over clustering is done, we always get homogeneous clusters.

Table 3.3. Results for the Iris Dataset

$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
0.18	2.18	49.76	43.88
0.14	2.76	45.92	29.42
0.12	3.74	36.34	19.42
0.10	5.12	25.56	15.16

3.4.2 Wine Recognition Dataset

The results for the Wine Recognition dataset are shown in Table 3.5. We get more errors than randomly initialized FCM because the ants sometimes find only 2 clusters and the errors increase dramatically because of this under clustering. Also, when more than 4 clusters are found the errors shoot up.

Table 3.4. Results for the Iris Dataset Considering 2 Classes

$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
0.20	2.02	3.74	2.94
0.18	2.14	1.16	2.58
0.12	3.74	0	0.18

Table 3.5. Results for the Wine Recognition Dataset

$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
0.085	4.46	13.94	9.36
0.09	3.9	14.3	9.02
0.1	3.76	23.08	10.4

3.4.3 Glass Identification Dataset

This is perhaps the most difficult dataset to cluster. The results for the Glass Identification dataset are shown in Table 3.6. If we allow over clustering then the errors are a little better than the randomly initialized FCM, but as we close in on the correct number of clusters, sometimes under clustering occurs and the errors shoot up.

The glass dataset can be considered as a two cluster dataset, *window glass* and *non-window glass*. The results for this modified dataset are shown in Table 3.7. For this dataset both over clustering and under clustering increase the errors, for this reason the errors for the ant algorithm are always greater than for randomly initialized FCM.

Table 3.6. Results for the Glass Identification Dataset

$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
0.075	9.36	102.38	88.2
0.085	8.08	103.76	93.34
0.105	5.04	111.52	99.72
0.12	4.54	115.72	102.9
0.16	2.56	136.82	118.94

Table 3.7. Results for the Glass Identification Dataset Considering 2 Classes

$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
0.12	4.54	22.52	21.26
0.135	2.62	40.64	25.36
0.15	2.22	49.7	27.3
0.16	2.56	49.84	26.22
0.18	2.16	49.96	28.04
0.20	1.78	50.12	31.58

3.4.4 Artificial Datasets

Ten artificial datasets, each with 2 attributes, were generated using a Gaussian distribution. The results for the datasets are shown in Tables 3.8–3.17.

For the Gauss-1, Gauss-2 and Gauss-3 datasets, over clustering and exact clustering gives us perfect clustering, that is there are no errors. But the errors increase dramatically on under clustering.

The Gauss-4 dataset, with 5 clusters has two extrema one with 0 errors and one with 211 errors. When over clustering occurs we sometimes get 211 errors, because of this we don't get perfect clustering for this dataset.

The Gauss-5 dataset, with 5 clusters has 7 errors. Over clustering results in comparable errors, but under clustering dramatically increases the errors.

For the Gauss500-1 dataset over clustering and exact clustering results in 1 error, but because of under clustering the errors shoot up.

For the Gauss500-2 dataset we get perfect clustering if we over cluster or get an exact clustering, but under clustering increases the errors.

For the Gauss500-3 dataset, over clustering doesn't dramatically increase the errors, but under clustering does. Also, we don't get perfect clustering for this dataset.

For the Gauss500-4 and Gauss500-5 datasets we get perfect clustering if we over cluster or get an exact clustering, but because of under clustering the errors shoot up.

Table 3.8. Results for the Gauss-1 Dataset

$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
0.19	8.26	130.26	0
0.22	7.44	181.14	9.5
0.235	6.56	268.16	2.64
0.25	5.84	348.58	25.34

Table 3.9. Results for the Gauss-2 Dataset

$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
0.19	9.76	27.08	0
0.225	7.96	136.92	0
0.25	6.96	212.68	5.28
0.28	5.86	369.68	15.84
0.295	4.60	492.08	118.14
0.30	4.48	560.24	136.08

Table 3.10. Results for the Gauss-3 Dataset

$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
0.25	7.44	191.9	0
0.265	6.56	276.18	13.2
0.28	5.66	380.96	46.98

Table 3.11. Results for the Gauss-4 Dataset

$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
0.19	6.32	318.3	32.18
0.225	6.1	327.08	40.62
0.25	6.42	328.04	27.96
0.27	6.24	327.42	33.76
0.28	5.86	327.96	37.98

Table 3.12. Results for the Gauss-5 Dataset

$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
0.19	8.1	188.42	8.06
0.205	6.8	252.96	10.34
0.215	6.3	329.4	20.56
0.22	5.94	375.28	58.22
0.25	4.4	552	119.84

Table 3.13. Results for the Gauss500-1 Dataset

$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
0.185	5.7	54.3	13.04
0.19	5.68	63.28	8.56
0.20	5	79.38	25

Table 3.14. Results for the Gauss500-2 Dataset

$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
0.17	6.1	9.76	0
0.19	5.64	37.36	7.6
0.21	5.04	75.68	30.4

Table 3.15. Results for the Gauss500-3 Dataset

$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
0.155	5.96	17.98	3.82
0.17	5.42	50.48	12.92
0.185	5.10	72.22	20.46
0.19	5.02	72.9	21.96

Table 3.16. Results for the Gauss500-4 Dataset

$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
0.19	6.36	1.48	0
0.195	6.34	3.62	0
0.22	5.18	62.3	32.14
0.25	4.02	153.18	90.8
0.28	3.94	175.16	97.3

Table 3.17. Results for the Gauss500-5 Dataset

$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
0.19	5.86	11.84	3.04
0.195	5.78	20.96	6.08
0.20	5.96	23.32	4.56

3.5 Effect of Parameter Variation

The above results show the sensitivity of the algorithm to the $T_{createforheap}$ parameter. The algorithm is not very sensitive to the other parameters. We show the results for the sensitivity of the algorithm to the number of iterations in Tables 3.18 – 3.19.

For the Iris dataset, the number of clusters found increases for two values of $T_{createforheap}$ and decreases for one value of $T_{createforheap}$. The number of clusters differ because the ants get more time to move and create new heaps or dump objects to existing heaps in the first stage, and merge heaps in the second stage. For the Wine dataset the number of clusters found decreases for all the different values of $T_{createforheap}$.

3.6 Variations in the Algorithm

The merging and creation of new heaps depend on the number of neighbors of the ant. If the number of neighbors increases then the ant has a higher probability of finding a heap among its neighbors. If we consider a 3D board, instead of a 2D board, the number of neighbors increases from 8 to 26, this gives the ant a higher probability of finding a heap.

Table 3.18. Effect of Variation of Iterations on Iris Dataset

$T_{create_forheap}$	Iterations	Classes found	Errors after ant stage	Errors after FCM
0.10	1000	5.12	25.56	15.16
0.10	2000	5.14	26.10	14.26
0.10	3000	5.30	26.56	13.68
0.12	1000	3.74	36.34	19.42
0.12	2000	3.64	35.76	17.68
0.12	3000	3.78	36.50	18.50
0.14	1000	2.76	45.92	29.42
0.14	2000	2.82	46.54	33.36
0.14	3000	2.92	45.14	27.76
0.18	1000	2.18	49.76	43.88
0.18	2000	2.12	49.90	46.24
0.18	3000	2.10	49.92	46.92

Table 3.19. Effect of Variation of Iterations on Wine Dataset

$T_{create_forheap}$	Iterations	Classes found	Errors after ant stage	Errors after FCM
0.085	1000	4.46	13.94	9.36
0.085	2000	4.38	13.88	9.52
0.085	3000	4.22	14.88	9.38
0.09	1000	3.9	14.3	9.02
0.09	2000	3.8	15.48	10.32
0.09	3000	3.54	15.84	8.86
0.10	1000	3.76	23.08	10.04
0.10	2000	3.56	25.52	11.52
0.10	3000	3.46	26.76	10.16

We tried two approaches for the 3D board, in the first approach, we considered a 3D board with $z=3$, that is the 3D board can be considered as three separate 2D boards. The ants can move from one 2D board to the other. In the second approach, all the three dimensions are equal. A point to be noted is that as the z -dimension increases, the size of the corresponding 2D board decreases.

The results are shown in Tables 3.20 – 3.29. The values of all the parameters, except the board dimension, are the same as the original 2D board. For all the datasets, except Iris (2 class), the

number of clusters found by using the 3D board is less than that found by using the 2D board. This is to be expected, because as the dimension increases, the number of neighbors increase, which in turn increases the probability of finding a heap. For the Gauss500 datasets, except Gauss500-3 dataset, the 3D board based algorithm always finds perfect clustering, which was not possible with the 2D board.

For 5 out of the 10 datasets, the 3D board with $z=3$ found fewer clusters than the 3D board with equal dimensions. For the remaining 5 datasets, most of the time the 3D board with $z=3$ found fewer clusters. For the Gauss500-3 and the Gauss500-5 datasets the 3D board with equal dimensions found fewer clusters than the 3D board with $z=3$.

Table 3.20. Results for the Iris Dataset (3D Board)

Board type	$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
2D	0.12	3.74	36.34	19.42
3D($z=3$)	0.12	3.38	36.38	22.56
3D($x=y=z$)	0.12	3.46	36.92	20.4
2D	0.14	2.76	45.92	29.42
3D($z=3$)	0.14	2.72	45.78	30
3D($x=y=z$)	0.14	2.72	46.92	30.08
3D($z=3$)	0.13	2.84	43.14	31.32
3D($x=y=z$)	0.13	3.06	43.16	25.28

Table 3.21. Results for the Iris Dataset (2 Class) (3D Board)

Board type	$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
2D	0.12	3.74	0	0.18
3D($z=3$)	0.12	3.38	0	0.42
3D($x=y=z$)	0.12	3.46	0	0.24
2D	0.18	2.14	1.16	2.58
3D($z=3$)	0.18	2.06	0.74	2.82
3D($x=y=z$)	0.18	2.1	0.7	2.7
2D	0.2	2.02	3.74	2.94
3D($z=3$)	0.2	2.02	4.44	3.82
3D($x=y=z$)	0.2	2.12	3.78	2.64

Table 3.22. Results for the Wine Dataset (3D Board)

Board type	$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
2D	0.085	4.46	13.94	9.36
3D(z=3)	0.085	3.68	14.26	8.78
3D(x=y=z)	0.085	3.8	16.00	8.76
2D	0.09	3.9	14.30	9.02
3D(z=3)	0.09	3.2	15.26	8.86
3D(x=y=z)	0.09	3.24	16.44	8.76
2D	0.1	3.76	23.08	10.04
3D(z=3)	0.1	3.16	25.02	12.58
3D(x=y=z)	0.1	3.12	26.66	10.12

Table 3.23. Results for the Glass Dataset (3D Board)

Board type	$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
2D	0.075	9.36	102.38	88.2
3D(z=3)	0.075	8.7	104.22	90.98
3D(x=y=z)	0.075	8.92	103.82	89.58
2D	0.08	8.08	103.76	93.34
3D(z=3)	0.085	6.42	106.46	95.36
3D(x=y=z)	0.085	6.44	106.44	95.74
2D	0.105	5.04	111.52	99.72
3D(z=3)	0.105	3.9	112.6	107.42
3D(x=y=z)	0.105	3.96	112.56	107.12

Table 3.24. Results for the Glass (2 Class) Dataset (3D Board)

Board type	$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
2D	0.12	4.54	22.52	21.26
3D(z=3)	0.13	2.92	33.7	23.36
3D(x=y=z)	0.13	2.84	32.56	23.46
3D(z=3)	0.135	2.48	40.36	23.74
3D(x=y=z)	0.135	2.6	38.36	24.22
3D(z=3)	0.14	2.1	47.3	28.96
3D(x=y=z)	0.14	2.26	46.14	29.22
2D	0.15	2.22	49.7	27.3
3D(z=3)	0.15	1.64	50.5	33.76
3D(x=y=z)	0.15	1.86	50.28	31.7
2D	0.16	2.56	49.84	26.22
3D(z=3)	0.16	1.56	50.86	35
3D(x=y=z)	0.16	1.84	50.42	32.32
2D	0.18	2.16	49.96	28.04
3D(z=3)	0.18	1.24	50.88	44.92
3D(x=y=z)	0.18	1.3	50.62	44.42

Table 3.25. Results for the Gauss500-1 Dataset (3D Board)

Board type	$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
3D(z=3)	0.17	4.98	26.76	2.5
3D(x=y=z)	0.17	5	23.66	1
3D(z=3)	0.175	4.94	31.24	5.5
3D(x=y=z)	0.175	4.98	30.72	4
3D(z=3)	0.18	4.8	43.18	16
3D(x=y=z)	0.18	4.9	39.44	8.5
2D	0.185	5.7	54.3	13.04
3D(z=3)	0.185	4.66	51.92	28
3D(x=y=z)	0.185	4.86	45	11.5
2D	0.19	5.68	63.28	8.56
3D(z=3)	0.19	4.62	59.42	29.5
3D(x=y=z)	0.19	4.64	56.52	28
2D	0.2	5	79.38	25
3D(z=3)	0.2	4.26	76.94	56.5
3D(x=y=z)	0.2	4.24	77.84	58

Table 3.26. Results for the Gauss500-2 Dataset (3D Board)

Board type	$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
2D	0.17	6.1	9.76	0
3D(z=3)	0.17	5	9.58	0
3D(x=y=z)	0.17	5	8.18	0
2D	0.19	5.64	37.36	7.6
3D(z=3)	0.19	4.7	37.42	22.8
3D(x=y=z)	0.19	4.76	35.44	19.76
2D	0.21	5.04	75.68	30.4
3D(z=3)	0.21	4.1	74.72	70.36
3D(x=y=z)	0.21	4.18	74.46	63.84

Table 3.27. Results for the Gauss500-3 Dataset (3D Board)

Board type	$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
3D(z=3)	0.155	4.92	15.98	7
3D(x=y=z)	0.155	4.92	16.5	7
3D(z=3)	0.16	4.84	23.88	13
3D(x=y=z)	0.16	4.8	27.1	16
3D(z=3)	0.165	4.66	32.62	26.5
3D(x=y=z)	0.165	4.58	40.12	32.5
2D	0.17	5.42	50.48	12.92
3D(z=3)	0.17	4.48	46.08	40
3D(x=y=z)	0.17	4.48	48.9	40
2D	0.185	5.1	72.22	20.46
3D(z=3)	0.185	4.08	73.2	70
3D(x=y=z)	0.185	4.06	73.22	71.5
2D	0.19	5.02	72.9	21.96
3D(z=3)	0.19	4	76.08	76
3D(x=y=z)	0.19	4.06	75.82	71.5

Table 3.28. Results for the Gauss500-4 Dataset (3D Board)

Board type	$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
2D	0.195	6.34	3.62	0
3D(z=3)	0.195	5	4.1	0
3D(x=y=z)	0.195	5	3.66	0
2D	0.22	5.18	62.3	32.14
3D(z=3)	0.22	4.38	66.74	59.72
3D(x=y=z)	0.22	4.62	42.44	36.02
3D(z=3)	0.235	3.8	116.28	110.52
3D(x=y=z)	0.235	4	101.72	94.48
2D	0.25	4.02	153.18	90.8

Table 3.29. Results for the Gauss500-5 Dataset (3D Board)

Board type	$T_{createforheap}$	Classes found	Errors after ant stage	Errors after FCM
3D($z=3$)	0.18	5.02	2.4	0
3D($x=y=z$)	0.18	5	4.32	0
3D($z=3$)	0.185	4.98	6.6	3.04
3D($x=y=z$)	0.185	4.98	7.3	1.52
2D	0.19	5.86	11.84	3.04
3D($z=3$)	0.19	4.96	10.46	4.56
3D($x=y=z$)	0.19	4.92	11.88	7.6
2D	0.195	5.78	20.96	6.08
2D	0.2	5.96	23.32	4.56
3D($z=3$)	0.2	4.7	29.64	22.8
3D($x=y=z$)	0.2	4.78	23.84	16.72
3D($z=3$)	0.215	4.1	79.3	70.5
3D($x=y=z$)	0.215	4.1	85.08	70.5

3.7 Discussion

The use of ants for 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. 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 found a systematic way to set the values of the parameters. The final partition found with 3 classes for the Iris dataset is always equivalent to what we get with FCM.

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 3.30 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.

The results for the Iris dataset, when considered as a two cluster dataset, were better than the randomly initialized FCM algorithm because sometimes over clustering occurs and we get perfect clustering. Also the ants find a better partition than FCM, because the ants are able to split off the linearly separable clusters.

Also, we found that the clustering was highly sensitive to $T_{createforheap}$. $T_{createforheap}$ clearly has a strong influence on how many final clusters are obtained.

In [4], 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 cluster centers. 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.

Another avenue we have pursued is to allow the ants to relocate clusters centroids in features space. The formulation is the same at a very high-level as was done in [14], but ants are utilized rather than a genetic approach.

Table 3.30. Typical Example of Variance in Data Partitions Obtained with the Wine Dataset with $T_{createforheap} = 0.09$ and 3000 Iterations

Clusters Found	Frequency	Errors after Ant stage	Errors after FCM
3	13	8.9230	9.00
4	18	14.6111	8.00
5	15	17.3333	11.00
6	3	24.00	11.00
7	1	33.00	10.00

CHAPTER 4

AUTOMATICALLY SETTING THE PARAMETERS

The ant algorithm is very sensitive to the threshold for deciding when to merge heaps and remove the items from the heap. Results from the previous chapter show the sensitivity of the algorithm to the $T_{createforheap}$ parameter. The algorithm is less sensitive to the other parameters.

4.1 New Metric for Merging Heaps

We tried different approaches to automatically set the $T_{createforheap}$ parameter.

The original approach for merging heaps is

If the distance between the centroid of the heap carried by the ant and the centroid of the heap on the board is less than a constant $T_{createforheap}$ then the heaps are merged.

Ants drop a heap H_1 onto another heap H_2 provided that:

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

The drawback of this metric is that there is no heuristic to set the value of $T_{createforheap}$, the value is determined empirically.

The new metric for merging heaps is

If the distance between the centroid of the heap carried and the centroid of the heap on the board is less than a fixed percent of the mean distance of all the objects in the heap from the center of the heap on the board then the heaps are merged.

Ants merge a heap H_1 onto another heap H_2 provided that:

$$D(O_{center}(H_1), O_{center}(H_2)) < percent \times D_{mean}(H_2)$$

The aim is to get a percentage value which is uniform across all the data sets.

4.2 Results Using the New Metric

The new metric was applied to the Iris plant dataset and the Wine recognition dataset. The results are shown in Tables 4.1 and 4.2. The results are encouraging on individual datasets but

the same *percentage value* doesn't work across the datasets. For the Iris dataset, 2.40 is the *best percentage value*. But for the wine dataset the value of 2.40 is too high, it results in only one cluster. The value of 1.25 is the best for the wine dataset.

Table 4.1. Results from the New Metric for Iris Dataset

<i>Percent</i>	Iterations	Classes found	Errors after Ant Stage	Errors after FCM
2.25	1000	4.18	47.16	21.14
2.30	1000	3.72	47.02	22.08
2.35	1000	3.56	47.64	21.68
2.40	1000	3.28	49.22	25.12
2.45	1000	2.88	49.70	29.90

Table 4.2. Results from the New Metric for Wine Dataset

<i>Percent</i>	Iterations	Classes found	Errors after Ant Stage	Errors after FCM
2.40	1000	1	107	107
2.00	1000	1	107	107
1.25	1000	3.44	31.18	22.06
1.00	1000	4.62	13.12	9.42

The effect of varying the number of iterations, using the new metric, is shown in Tables 4.3 and 4.4. For both the Iris and Wine datasets as the number of iterations increase, the clusters found decrease. This is to be expected, because as the number of iterations increase the ants get more time to search the board and merge the heaps.

Table 4.3. Effect of Variation of Iterations on Iris Dataset

<i>Percent</i>	Iterations	Classes found	Errors after Ant Stage	Errors after FCM
2.25	1000	4.18	47.16	21.14
2.25	4000	3.48	48.94	21.60
2.30	1000	3.72	47.02	22.08
2.30	4000	3.22	49.16	23.42
2.35	1000	3.56	47.64	21.68
2.35	4000	3.12	49.28	28.20
2.40	1000	3.28	49.22	25.12
2.40	3000	2.90	49.52	29.36
2.40	4000	2.94	49.40	29.40
2.45	1000	2.88	49.70	29.90
2.45	4000	2.60	51.00	33.42

Table 4.4. Effect of Variation of Iterations on Wine Dataset

<i>Percent</i>	Iterations	Classes found	Errors after Ant Stage	Errors after FCM
2.45	1000	1.00	107.00	107
2.45	3000	1.00	107.00	107
2.00	1000	1.00	107.00	107
2.00	3000	1.00	107.00	107
1.25	1000	3.44	31.18	22.06
1.25	3000	3.04	37.24	28.28
1.00	1000	4.62	13.12	9.42
1.00	3000	4.32	14.40	9.26

4.3 Fuzzy Hypervolume

Traditional algorithms for determining the number and structure of clusters incorporate some form of *Partition Entropy* [9, 13, 25, 32, 21, 22, 24, 33]. The membership matrix U is used to compute the entropy of the partition. The membership of the object in cluster u_{ij} is interpreted as the probability for the object j to be in cluster i [25]. The number of clusters is varied and the

partition entropy is computed. The optimum number of clusters is the one corresponding to the minimum partition entropy.

In [12] the authors have used the Fuzzy Hypervolume and density criteria to find the number of clusters. The exponential distance measure, $d_e^2(X_j, V_i)$, based on the maximum likelihood estimation is used. The exponential distance is used in calculation of the posterior probability of selecting the i^{th} cluster given the j^{th} feature vector $h(i|X_j)$.

$$h(i|X_j) = \frac{\frac{1}{d_e^2(X_j, V_i)}}{\sum_{k=1}^K \frac{1}{d_e^2(X_j, V_k)}} \quad (4.1)$$

$$d_e^2(X_j, V_i) = \frac{[\det(F_i)]^{\frac{1}{2}}}{P_i} \exp \left[\frac{(X_j - V_i)^T F_i^{-1} (X_j - V_i)}{2} \right] \quad (4.2)$$

where

- F_i : Fuzzy covariance matrix of the i^{th} cluster
- P_i : *a priori* probability of selecting the i^{th} cluster

The fuzzy covariance matrix of the i^{th} cluster is computed as:

$$F_i = \frac{\sum_{j=1}^N h(i|X_j) (X_j - V_i)(X_j - V_i)^T}{\sum_{j=1}^N h(i|X_j)} \quad (4.3)$$

The Fuzzy Hypervolume is defined as:

$$F_{HV} = \sum_{i=1}^K [\det(F_i)]^{\frac{1}{2}} \quad (4.4)$$

The fuzzy hypervolume of the datasets, assuming uniform prior probabilities, was computed to find a basis of setting the $T_{createforheap}$ parameter. The hypothesis is that there is a correlation between $T_{createforheap}$ and the fuzzy hypervolume. $T_{createforheap}$ could be modeled as a function of the hypervolume and could be automatically set at run time. The fuzzy hypervolume for the datasets with the best value found for $T_{createforheap}$ is shown in Table 4.5.

Table 4.5. $T_{createforheap}$ and Fuzzy Hypervolume

Dataset	$T_{createforheap}$	Fuzzy Hypervolume	Fuzzy Hypervolume(log)
Iris	0.12	3.5186E-04	-3.453633681
Wine	0.09	1.5980E-11	-10.79641912
Glass	0.085	5.7413E-10	-9.240991303
Gauss-1	0.235	9.4408E-02	-1.024993149
Gauss-2	0.25	1.0947E-01	-0.960688101
Gauss-3	0.265	1.0682E-01	0.971361128
Gauss-4	0.25	1.0775E-01	-0.967596587
Gauss-5	0.21	7.1804E-02	-1.143849094
Gauss500-1	0.2	7.8007E-02	-1.107865773
Gauss500-2	0.19	9.0652E-02	-1.0426242
Gauss500-3	0.185	8.5193E-02	-1.069593575
Gauss500-4	0.195	8.6286E-02	-1.064060967
Gauss500-5	0.195	7.9956E-02	-1.097149734

The plots of $T_{createforheap}$ vs Fuzzy Hypervolume and $T_{createforheap}$ vs Fuzzy Hypervolume (log) are shown in Figure 4.1 and 4.2. From the above results we can conclude that the two parameters are not correlated.

Figure 4.1. $T_{createforheap}$ vs Fuzzy Hypervolume

Figure 4.2. $T_{createforheap}$ vs Fuzzy Hypervolume (Log)

CHAPTER 5

FUZZY ANT CLUSTERING WITH CENTROIDS

5.1 Introduction

Previous ant based clustering algorithms cluster data by moving the objects in a 2D space and merging them to form clusters. Similar objects tend to merge together to form heaps. This merging is controlled by a threshold which dictates the permissible dissimilarity between the objects in a cluster. The ant based clustering algorithms are very sensitive to this threshold [19]. The algorithm presented in Chapter 3 finds the number of cluster centers and good initial cluster centers for the Fuzzy C Means algorithm. The algorithm is very sensitive to the $T_{createforheap}$ parameter.

In the proposed algorithm the stochastic property of ants was simulated to obtain good cluster centers. The ants move randomly in the feature space carrying a feature of a cluster center with them. After a fixed number of iterations the cluster centers are evaluated using the reformulation of FCM which leaves out the membership matrix [15]. After the ant stage the best cluster centers obtained are used as the initial cluster centers for the Fuzzy C Means and Hard C Means algorithms.

5.2 Reformulation of Clustering Criteria for FCM and HCM

In [15] the authors have proposed a reformulation of the optimization criteria used in a couple of common clustering objective functions. The original clustering functions minimize the objective function 5.1 to find good clusters.

$$J_m(U, \beta) = \sum_{i=1}^c \sum_{k=1}^n U_{ik}^m D_{ik}(x_k, \beta_i) \quad (5.1)$$

where

- $c \geq 2$: Number of clusters
- n : Number of data points
- β_i : The i^{th} cluster prototype
- $D_{ik}(x_k, \beta_i)$: Distance of x_k from i^{th} cluster center

- U_{ik} : Membership of the k^{th} object in the i^{th} cluster
- $m \geq 1$: The degree of fuzzification

The reformulation replaces the membership matrix U with the necessary conditions which are satisfied by U . The reformulated version of J_m is denoted as R_m .

For the Hard clustering case the U optimization is over a crisp membership matrix. The necessary condition for U is given in equation 5.2. Equation 5.3 gives the the necessary conditions for U , for the fuzzy case. The distance $D_{ik}(x_k, \beta_i)$ is denoted as D_{ik} .

$$\begin{aligned} U_{ik} &= 0 \text{ if } D_{ik} > \min (D_{1k}, D_{2k}, D_{3k}, \dots, D_{ck}) \\ &= 1 \text{ otherwise} \end{aligned} \quad (5.2)$$

$$U_{ik} = \frac{\left(D_{ik}^{\frac{1}{1-m}}\right)}{\left(\sum_{j=1}^c D_{jk}^{\frac{1}{1-m}}\right)} \quad (5.3)$$

The reformulations for hard and fuzzy optimization functions are given in equations 5.4 and 5.5 respectively. The function R depends only on the cluster prototype and not on the U matrix, whereas J depends on both the cluster prototype and the U matrix. The U matrix for the reformulated criterion can be easily computed using equation 5.2 or 5.3.

$$R_1(\beta) = \sum_{k=1}^n \min (D_{1k}, D_{2k}, \dots, D_{ck}) \quad (5.4)$$

$$R_m(\beta) = \sum_{k=1}^n \left(\sum_{i=1}^c D_{ik}^{\frac{1}{1-m}} \right)^{1-m} \quad (5.5)$$

5.3 Algorithm

The ants co-ordinate to move cluster centers in feature space to search for optimal cluster centers. Initially the feature values are normalized between 0 and 1. Each ant is assigned to a particular feature of a cluster in a partition. The ants never change the feature, cluster or partition assigned to them. A pictorial view is given in Figure 5.1 where each vertical line is a dimension in parallel coordinates [17, 2]. After randomly moving the cluster centers for a fixed number of iterations,

called an epoch, the quality of the partition is evaluated by using the reformulated criterion 5.4 or 5.5. If the current partition is better than any of the previous partitions' in the ant's memory then the ant remembers this partition else the ant, with a given probability goes back to a better partition or continues from the current partition. This ensures that the ants do not remember a bad partition and erase a previously known good partition. Even if the ants change *good* cluster centers to *unreasonable* cluster centers, the ants can go back to the *good* cluster centers as the ants have a finite memory in which they keep the currently best known cluster centers. There are two directions for the random movement of the ant. The positive direction is when the ant is moving in the feature space from 0 to 1, and the negative direction is when the ant is moving in the feature space from 1 to 0. If during the random movement the ant reaches the end of the feature space the ant reverses direction. After a fixed number of epochs the ants stop.

The data is partitioned using the centroids obtained from the best known R_m value. The nearest neighbor algorithm is used for assignment to a cluster. The cluster centers so obtained are then used as the initial cluster centers for the Fuzzy C Means or the Hard C Means algorithm. The ant based algorithm is presented in Figure 5.2.

Figure 5.1. Pictorial View of the Algorithm

1. Normalize the feature values between 0 and 1. The normalization is linear. The minimum value of a particular feature is mapped to 0 and the maximum value of the feature is mapped to 1.
2. Initialize the ants with random initial values and with random direction. There are two directions, positive and negative. The positive direction means the ant is moving in the feature space from 0 to 1. The negative direction means the ant is moving in the feature space from 1 to 0. Clear the initial memory. The ants are initially assigned to a particular feature within a particular cluster of a particular partition. The ants never change the feature, cluster or the partition assigned to them.
3. Repeat
 - 3.1 For one epoch /* One epoch is n iterations of random ant movement */
 - 3.1.1 For all ants
 - 3.1.1.1 With a probability P_{rest} the ant rests for this epoch
 - 3.1.1.2 If the ant is not resting then with a probability $P_{continue}$ the ant continues in the same direction, else it changes direction
 - 3.1.1.3 With a value between D_{min} and D_{max} the ant moves in the selected direction
 - 3.2 The new R_m value is calculated using the new cluster centers
 - 3.2.1 If the partition is better than any of the old partitions in memory then the worst partition is removed from the memory and this new partition is copied to the memories of the ants making up the partition
 - 3.2.2 If the partition is not better than any of the old partitions in memory

Then

With a probability $P_{ContinueCurrent}$ the ant continues with the current partition

Else

With a probability 0.6 the ant chooses to go back to the best known partition, with a probability 0.2 the ant goes back to the second best known partition, with a probability 0.1 the ant goes to the third best known partition, with a probability 0.075 the ant goes to the fourth best known partition and with a probability 0.025 the ant goes to the worst known partition

Until Stopping criteria

The stopping criterion is the number of epochs.

Figure 5.2. Fuzzy Ant Clustering with Centroids Algorithm

The values of the parameters used in the algorithm are shown in Table 5.1.

Table 5.1. Parameter Values

Parameter	Value
Number of ants	30 Partitions ¹
Memory per ant	5
Iterations per epoch	50
Epochs	1000
P_{rest}	0.01
$P_{continue}$	0.75
$P_{ContinueCurrent}$	0.20
D_{min}	0.001
D_{max}	0.01

5.4 Results

The algorithm was applied to six real datasets and ten artificial datasets: the Iris Plant dataset, Glass Identification dataset, Wine Recognition dataset, MRI dataset, Multiple Sclerosis dataset, the British Towns dataset and the 10 Gaussian datasets.

The results obtained for the datasets are shown in Table 5.2. All results are an average from 50 random initializations. The results for the FCM and HCM are the average results from 50 random initializations. The glass dataset has been simplified to have just 2 classes *window glass* and *non-window glass*. The results for this modified dataset are also shown in Table 5.2. The age factor plays an important role in the Multiple Sclerosis dataset; the results considering the age feature and ignoring the age feature are also shown. Note, the R_m value is always less than or equal to that from randomly initialized FCM except for Glass (6 classes). Thirteen datasets have a single extrema for the FCM algorithm. They converge to the same extrema, for all initializations tried here. This is reflected in Table 5.2 where we have the same values in columns 3 and 4 for the thirteen datasets.

The parameters *Number of epochs*, D_{min} and D_{max} play an important role in determining the quality of the clusters found. By performing manual search, new parameters, which gave better results, were found. The values of the new parameters are shown in Table 5.3 and the results obtained by using these modified parameters are shown in Table 5.4. Clear improvements were observed for 3 datasets. For the British Towns dataset the average value for R_m after FCM

¹1 partition = number of clusters $\times$ number of features per cluster

Table 5.2. Results for Fuzzy C Means

Dataset	Min R_m found by Ants	R_m from FCM, Ant Initialization	R_m from FCM, Random Initialization
British Towns	1.6828	1.6033	1.6039
Iris	5.4271	5.2330	5.2330
Wine	33.0834	28.7158	28.7158
Glass (6 classes)	11.3827	7.2937	7.2917
Glass (2 classes)	25.8531	24.3932	24.3932
Multiple Sclerosis (with age)	6.9456	6.8538	6.8538
Multiple Sclerosis (ignoring age)	3.5704	3.5319	3.5319
MRI	311.2397	302.1302	303.289
Gauss-1	6.1588	5.5481	5.5481
Gauss-2	4.6786	4.0646	4.0646
Gauss-3	4.8764	4.2120	4.2120
Gauss-4	2.5536	1.9156	2.7458
Gauss-5	8.6553	8.2035	8.2035
Gauss500-1	4.4866	4.2559	4.2559
Gauss500-2	3.6328	3.3681	3.3681
Gauss500-3	2.3564	2.0560	2.0560
Gauss500-4	1.9937	1.6880	1.7834
Gauss500-5	1.9031	1.5848	1.5848

decreased to 1.5999 from 1.6033, similarly for the Glass (6 classes) dataset the average value for R_m after FCM decreased to 7.2897 from 7.2937. This average value is better than that obtained from randomly initialized FCM. The average value for R_m after FCM for the MRI dataset decreased to 301.9198 from 302.1302.

Table 5.3. New Parameters

Parameter	Old Value	New Value
Epochs	1000	2000
D_{min}	0.001	0.0001
D_{max}	0.01	0.001

Table 5.4. Results for FCM Obtained from Modified Parameters

Dataset	Min R_m found by Ants	R_m from FCM, Ant Initialization	R_m from FCM, Random Initialization
British Towns	1.6051	1.5999	1.6039
Glass (6 classes)	9.2284	7.2897	7.2917
MRI	302.1188	301.9189	303.2894

5.5 Hard C Means

The ant algorithm was applied together with the Hard C Means algorithm. The ants find the cluster centers and these centers are used as the initial centers for the Hard C Means algorithm. The parameter values are those shown in Table 5.1.

From Table 5.5 we see that the algorithm gives better results than randomly initialized HCM for 15 of the 18 datasets tested. Changing the parameter values can improve the results. By performing a search in the parameter space we got parameter values that resulted in better partitions. Tables 5.6 and 5.7 show the variation in the results obtained by changing the number of ants per partition and epochs for the British Towns' and Wine datasets. From the tables we see that as the number of epochs increase, the minimum R_m found by the ants decreases, this is to be expected because as the number of epochs increase, the ants get more time to refine the centroids found. Also as the number of ants increase, better R_m values are found. Table 5.8 shows the results obtained for different MRI slices, the parameter values used for the MRI dataset are tabulated in Table 5.3 and the ants per partition were 50. We can see the partitions all have lower R_m values.

Table 5.5. Results for Hard C Means

Dataset	Min R_m found by Ants	R_m from HCM, Ant Initialization	R_m from HCM, Random Initialization
British Towns	5.5202	3.6260	3.4339
Iris	7.0055	6.9981	8.2516
Wine	52.8098	50.4573	48.9792
Glass (6 classes)	28.1317	24.3770	21.1165
Glass (2 classes)	34.2488	34.1352	36.9132
Multiple Sclerosis (with age)	10.2201	10.2016	10.3548
Multiple Sclerosis (ignoring age)	4.6406	4.6381	4.7759
MRI	433.2752	432.7499	452.384
Gauss-1	7.1391	6.4856	11.0962
Gauss-2	5.1055	4.4725	11.0645
Gauss-3	5.2625	4.6624	12.7167
Gauss-4	2.7273	2.0386	12.4133
Gauss-5	11.6107	10.9422	11.9558
Gauss500-1	5.8333	5.4921	6.0718
Gauss500-2	4.3405	4.0029	7.7512
Gauss500-3	2.6558	2.3140	6.3784
Gauss500-4	2.1678	1.8465	7.3442
Gauss500-5	2.0758	1.7314	7.0953

Table 5.6. Results for the British Towns Dataset

Ants per partition	Epochs	Min R_m found by Ants	R_m from HCM, Ant Initialization	R_m from HCM, Random Initialization
50	2000	5.3658	3.5812	3.4339
50	4000	4.5048	3.5701	
75	2000	5.1691	3.6134	
100	2000	3.0835	3.0661	

Table 5.7. Results for the Wine Dataset

Ants per partition	Epochs	Min R_m found by Ants	R_m from HCM, Ant Initialization	R_m from HCM, Random Initialization
50	2000	52.8003	49.2405	48.9792
50	4000	51.0631	49.2604	
75	2000	50.1879	49.2604	
75	3000	49.9230	48.9748	
100	2000	49.6415	48.9716	
100	4000	49.2076	48.9697	

Table 5.8. Results for the MRI Dataset

Slice #	Min R_m found by Ants	R_m from HCM, Ant Initialization	R_m from HCM, Random Initialization
20	853.7991	851.8342	882.5732
35	919.8082	917.6636	927.6961
45	839.1622	838.0175	851.3756
46	842.5583	841.3414	847.0730
47	796.8415	795.6057	834.1028

5.6 Execution Time

The variation of the minimum R_m found by the ants by changing the ants per partition for MR slice # 35 is shown in Table 5.9. As the number of ants increases the minimum R_m found decreases, but at the cost of increased execution time. As the number of ants increase, more search space is explored and we get better R_m values.

Table 5.9. Variation of R_m with the Number of Ants for Slice # 35 of MRI Dataset

Ants per partition	Min R_m found by Ants	R_m from HCM, Ant Initialization	R_m from HCM, Random Initialization
50	919.8082	917.6636	927.6961
75	912.0365	909.9324	
100	910.0054	907.9996	

The execution time for the British Towns and the MRI dataset is shown in Table 5.10. These two datasets were chosen because British Towns' dataset is the smallest (in terms of number of examples) dataset and the MRI dataset is the largest dataset used in the study. The values are an average from 20000 epochs and 5 experiments for the British Towns' dataset and from 6000 epochs and 3 experiments for the MRI dataset. One experiment consists of the ant stage and the following Fuzzy C Means or the Hard C Means stage. The time required for one epoch for the British towns dataset was more than that for the MRI dataset as there are more classes, and hence more ants, in the British towns dataset. The time required for the entire experiment was more for the MRI dataset as there are more examples in the MRI dataset. The time required for the FCM algorithm is also shown in Table 5.10. From Table 5.10 we see that, for the British Towns' dataset, approximately 9100 random FCM initializations can be performed in the time required for one ant experiment. Similarly approximately 225 random FCM initializations can be performed for the MRI dataset.

We performed 11300 ($\approx 225 \times 50$) random initializations for the MRI dataset and averaged the best 50 R_m values. The average R_m value was 301.9189 and the average R_m value found by ants from 50 random initializations was 301.9189. Similarly for the British Towns', dataset the average of best 50 R_m values over 20000 iterations was 1.5999 and the average R_m value found by ants, from 50 random initializations, was 1.5999.

The experiments were performed on an Intel Pentium 4 2.53 GHz processor with 512 KB cache and 2 GB of main memory.

Table 5.10. Execution Time

Dataset	Ants per Partition	Time for one Epoch (milliseconds)	Time for one Experiment (seconds)	Time for FCM (seconds)
British Towns	50	14.9645	97	0.01318
	75	22.2210	108.8380	
	100	29.7060	120.896	
MRI	50	6.7333	540.72	2.3929
	75	10.1117	811.0133	

5.7 Conclusions

The algorithm is based on relocating the cluster centroids in the feature space. The ants move the cluster centers, not the objects, in feature space to find a good partition for the data. The algorithm does not use the object merging criterion, which makes it independent of the threshold for merging the objects. Also there are less controlling parameters than the previous ant based clustering algorithms [19].

Results from 18 datasets show the superiority of our algorithm over the randomly initialized FCM and HCM algorithms. For comparison purposes, Tables 5.11, 5.12 and 5.13 show the frequency of occurrence of different extrema for the ant initialized FCM and HCM algorithms and the randomly initialized FCM and HCM algorithms. The ant initialized FCM algorithm always finds the better extrema for the MRI dataset and for the British Towns' dataset the ant initialized algorithm finds the better extrema 49 out of 50 times. The ant initialized HCM algorithm finds the better extrema for the Iris dataset all the time and for the Glass (2 class) dataset majority of the time. For the different MRI slices, the ant initialized HCM algorithm finds the better extrema most of the time. In [14], a Genetic programming approach was used to optimize the clustering criteria, the genetic approach for Hard C Means, found better extrema 64% of the time for the Iris dataset. The ant initialized HCM finds better extrema all the time.

The number of ants per partition is an important parameter of the algorithm. The quality of the partition improves as number of ants increase, but the improvement comes at the expense of increased execution time. Future work should focus on automatically finding the number of ants per partition and the number of clusters. In this direction the algorithm proposed in [19] can be used to find the number of clusters.

Table 5.11. Frequency of Different Extrema from FCM, for British Towns and MRI Datasets

Dataset	Extrema	Frequency FCM, Ant Initialization	Frequency FCM Random Initialization
British Towns	1.5999	49	16
	1.6037	1	18
	1.6081	0	16
MRI	301.9195	50	37
	307.1898	0	13

Table 5.12. Frequency of Different Extrema from HCM, for Glass (2 Class) and Iris Datasets

Dataset	Extrema	Frequency HCM, Ant Initialization	Frequency HCM Random Initialization
Glass (2 class)	34.1320	19	3
	34.1343	11	19
	34.1372	19	15
	34.1658	1	5
Iris	6.9981	50	23
	7.1386	0	14
	10.9083	0	5
	12.1437	0	8

Table 5.13. Frequency of Different Extrema from HCM, MRI Dataset

Slice #	Extrema	Frequency HCM, Ant Initialization	Frequency HCM Random Initialization
20	841.3414	50	44
	889.1043	0	6
35	930.1677	45	30
	951.4871	5	15
	1003.492	0	5
45	838.0175	50	41
	912.2289	0	9
46	841.3414	50	45
	889.1043	0	5
47	795.3043	35	27
	796.2459	15	13
	970.5483	0	10

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Combined Algorithm

The *Fuzzy ants* algorithm automatically finds the number of clusters and good initial cluster centers. The drawback of the algorithm is the large number of controlling parameters and the absence of good methods to set those parameters.

The *Fuzzy ant clustering with centroids* algorithms has less controlling parameters and less sensitivity to the parameters. The drawback of the algorithm is it requires the initial knowledge of the number of clusters in data. The results are generally better than with FCM/HCM. One would need a large number of random initialization to be competitive. Also, a parallel version of the ants algorithm could operate much faster than the current sequential implementation, thereby making it a clear choice for minimizing the chance of finding a poor extrema.

The *Fuzzy ants* algorithm can be used to find the number of clusters in the data and the results of the algorithm can be given to the *Fuzzy ant clustering with centroids* algorithm.

We implemented the above idea and the results are promising. Tables 6.1 and 6.2 show the results from the combined algorithm. The analysis can be performed using two metrics: *Number of errors* and minimum R_m . For comparison, the results from randomly initialized FCM algorithm, with known number of clusters, are also given.

In the second ant stage the ants optimize the R_m criteria, this can be observed in Table 6.1 . The R_m value for the Iris dataset after the 1st stage FCM is 4.5220 and after the final FCM stage the value is 4.4738. But the minimum R_m value sometimes doesn't correspond to the minimum errors, this is observed in the Iris dataset, the errors after FCM 1st stage are 15.92 and those after the final FCM stage are 18.52.

Table 6.1. Results from the Combined Algorithm

Dataset	Classes found	R_m after FCM 1 st stage	R_m found by Ants (2 nd stage)	R_m from FCM (final)	R_m from FCM (random)
Iris	3.64	4.5220	4.7658	4.4738	5.2330
Wine	4.46	20.5878	22.9632	20.5878	28.7158
Glass	5.5	8.4098	9.9975	8.4098	7.2917
British Towns	6.74	1.1452	1.2429	1.1453	1.6039
Multiple Sclerosis	2.76	4.6844	4.8632	4.6844	6.8538
Gauss-1	5.84	6.2395	6.6336	6.1234	5.5481
Gauss-2	5.86	4.4591	5.1016	4.3724	4.0646
Gauss-3	6.34	4.1108	4.7385	4.1313	4.2120
Gauss-4	6.16	2.0323	2.7307	2.0462	2.7458
Gauss-5	6.30	7.1248	7.5398	7.0827	8.2035
Gauss500-1	5.68	4.0874	4.2854	4.0554	4.2559
Gauss500-2	5.64	3.3677	3.7338	3.3395	3.3681
Gauss500-3	5.02	2.6648	2.8838	2.6754	2.05560
Gauss500-4	5.18	2.7631	3.0501	2.7611	1.7834
Gauss500-5	5.78	1.6849	1.9576	1.6513	1.5848

Table 6.2. Results from the Combined Algorithm (Errors)

Dataset	Classes found	Errors after Ants 1 st stage	Errors after FCM (1 st stage)	Errors after FCM (final)	Errors after FCM (Random)
Iris	3.64	35.76	15.92	18.52	16
Wine	4.46	13.94	9.36	9.36	9
Glass	5.5	109.8	98.02	98.02	96
Multiple Sclerosis	2.76	20.86	17.86	17.86	17
Gauss-1	5.84	348.58	25.34	29.08	0
Gauss-2	5.86	369.68	15.84	15.84	0
Gauss-3	6.34	318.44	10.56	18.24	0
Gauss-4	6.16	328.98	7.92	7.92	25.32
Gauss-5	6.30	329.40	19.64	20.12	7
Gauss500-1	5.68	63.28	8.5	16.22	1
Gauss500-2	5.64	37.36	7.6	11.48	0
Gauss500-3	5.02	72.9	19	18.98	1
Gauss500-4	5.18	62.3	22.12	29.88	1.94
Gauss500-5	5.78	20.96	6.08	6.08	0

6.2 Contributions

The *Fuzzy Ants* algorithm is a novel method to find the number of clusters in the data and also provides good initializations for the FCM and HCM algorithms. We performed sensitivity analysis on the controlling parameters and found the *Fuzzy Ants* algorithm to be very sensitive to the $T_{createforheap}$ parameter.

The *Fuzzy ant clustering with centroids* algorithms has less controlling parameters and less sensitivity to the parameters. The drawback of the algorithm is it requires the initial knowledge of the number of clusters in data. The results are generally better than with FCM/HCM. One would need a large number of random initialization to be competitive. Also, a parallel version of the ants algorithm could operate much faster than the current sequential implementation, thereby making it a clear choice for minimizing the chance of finding a poor extrema. The combined algorithm finds the number of clusters in data and good partition for the data.

6.3 Future Work

The future work should focus on setting the $T_{createforheap}$ parameter automatically. The algorithm is very sensitive to this parameter. We have tried a couple of approaches to set the parameter automatically, but the results were not encouraging.

In the *Fuzzy ant clustering with centroids* algorithm, the number of ants is an important parameter, and also the initial number of clusters in the data is required. Future work should focus on setting these values automatically. We compare the results from the algorithms with randomly initialized FCM and HCM. In future the results can be compared with cleverly initialized FCM and HCM algorithms. The initial placement of the ants is also random, future work should focus on giving clever initializations to the ants. The stopping criteria is currently based on the number of iterations, future work should concentrate on automatically stopping the ants based on their progress. Trails can be used to guide the movement of the ants, future work should concentrate on incorporating trails in the algorithm.

REFERENCES

- [1] A. Abraham, J. R. del Solar, and M. Köppen, editors. *Soft Computing Systems - Design, Management and Applications, HIS 2002*, volume 87 of *Frontiers in Artificial Intelligence and Applications*, Santiago, Chile, Dec 2002. IOS Press.
- [2] M. Berthold and L. Hall. Visualizing fuzzy points in parallel coordinates. *IEEE Transactions on Fuzzy Systems*, 11(3):369–374, June 2003.
- [3] J. C. Bezdek. Pattern recognition with fuzzy objective functions algorithms. *Plenum press*, 1981.
- [4] W. Bin, Z. Yi, L. Shaohui, and S. Zhongzhi. Csim: a document clustering algorithm based on swarm intelligence. *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, 1:477–482, May 2002.
- [5] W. Bin and S. Zhongzhi. A clustering algorithm based on swarm intelligence. *Info-tech and Info-net, 2001 Proceedings. ICII 2001*, 3:58–66, 2001.
- [6] C. Blake and C. Merz. UCI repository of machine learning databases, 1998.
- [7] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence From Natural to Artificial Systems*. Oxford University Press, New York, New York, 1999.
- [8] E. Bonabeau, A. Sobkowski, G. Theraulaz, and J.-L. Deneubourg. Adaptive task allocation inspired by a model of division of labor in social insects. In D. Lundh, B. Olsson, and A. Narayanan, editors, *Biocomputing and Emergent Computation*, pages 36–45. World Scientific, 1997.
- [9] C.-H. Cheng. *Entropy Based Subspace Clustering for Mining Numerical Data*. PhD thesis, Chinese University of Hong Kong, 1999.
- [10] M. Doriga, V. Maniezzo, and A. Colomi. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 26(1):29–41, Feb 1996.
- [11] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York, 1973.
- [12] Gath and A.B.Geva. Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):773–781, Jul 1989.
- [13] F. Golchin and K. Paliwal. Minimum-entropy clustering and its application to lossless image coding. *Proc. IEEE International Conference on Image Processing*, Oct 1997.
- [14] L. O. Hall, I. B. Özyurt, and J. C. Bezdek. Clustering with a genetically optimized approach. *IEEE Trans. on Evolutionary Computation*, 3(2):103–112, 1999.
- [15] R. J. Hathway and J. C. Bezdek. Optimization of clustering criteria by reformulation. *IEEE Transactions on Fuzzy Systems*, 3(2):241–245, May 1995.

- [16] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis*. John Wiley, West Sussex, England, 1999.
- [17] A. Inselberg. The plane with parallel coordinates. *Special Issue on Computational Geometry, The Visual Computer*, (1):69–97, 1985.
- [18] K. Jajuga, A. Sokołowski, and H.-H. Bock, editors. *International Federation of Classification Societies*, number 8th in Studies in Classification, Data Analysis, and Knowledge Organization, Carcow, Poland, Jul 2002. Springer-Verlag.
- [19] P. M. Kanade and L. O. Hall. Fuzzy ants as a clustering concept. *North American Fuzzy Information Processing Society, NAFIPS 2003, 22nd International Conference of the*, pages 227–232, 2003.
- [20] P. M. Kanade and L. O. Hall. Fuzzy ants clustering with centroids. *accepted for publication in FUZZ-IEEE 2004 conference*, 2004.
- [21] N. Karayiannis. Maximum entropy clustering algorithms and their application in image compression. *Systems, Man, and Cybernetics, 1994. 'Humans, Information and Technology', 1994 IEEE International Conference on*, 1:337–342, Oct 1994.
- [22] N. Karayiannis. Meca: maximum entropy clustering algorithm. *Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the Third IEEE Conference on*, 1:630–635, Jun 1994.
- [23] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, San Diego, California, 2001.
- [24] R.-P. Li and M. Mukaidono. A maximum-entropy approach to fuzzy clustering. *Fuzzy Systems, 1995. International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium., Proceedings of 1995 IEEE International Conference on*, 4:2227–2232, Mar 1995.
- [25] A. Lorette, X. Descombes, and J. Zerubia. Fully unsupervised fuzzy clustering with entropy criterion. *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, 3:986–989, Sep 2000.
- [26] N. Monmarché, M. Slimane, and G. Venturini. Antclass: discovery of clusters in numeric data by an hybridization of an ant colony with the kmeans algorithm, Jan 1999.
- [27] N. Monmarché, M. Slimane, and G. Venturini. On improving clustering in numerical databases with artificial ants. In D. Floreano, J. Nicoud, and F. Mondala, editors, *5th European Conference on Artificial Life (ECAL'99), Lecture Notes in Artificial Intelligence*, volume 1674, pages 626–635, Lausanne, Switzerland, Sep 1999. Springer-Verlag.
- [28] S. Quadfel and M. Batouche. Mrf-based image segmentation using ant colony system. *Electronic Letters on Computer Vision and Image Analysis*, 2(2):12–24, 2003.
- [29] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas. Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computing*, 6(4):321–332, Aug 2002.
- [30] V. Ramos and F. Almeida. Artificial ant colonies in digital image habitats - a mass behaviour effect study on pattern recognition. *Proc. of ANTS'2000 - 2nd Int. Workshop on Ant Algorithms (From Ant Colonies to Artificial Ants)*, pages 133–116, 2000.
- [31] V. Ramos and J. J. Merelo. Self-organized stigmergic document maps: Environment as mechanism for context learning. *Procs. of AEB02 - Spanish Conference on Evolutionary and BioInspired Algorithms*, pages 284–293, 2002.

- [32] S. Roberts, R. Everson, and I. Rezek. Minimum entropy data partitioning. *Proc. of 9th International Conference on Artificial Neural Networks*, pages 844–849, 1999.
- [33] D. Tran and M. Wagner. Fuzzy entropy clustering. *Fuzzy Systems, 2000. FUZZ IEEE 2000. The Ninth IEEE International Conference on*, 4:152–157, May 2000.
- [34] C.-F. Tsai, H.-C. Wu, and C.-W. Tsai. A new data clustering approach for data mining in large databases. *Parallel Architectures, Algorithms and Networks, 2002. I-SPAN '02 Proceedings. International Symposium on*, pages 278–283, 2002.
- [35] G. Weiss, editor. *Multiagent Systems A modern Approach to Distributed Artificial Intelligence*. MIT Press, Massachusetts, 1999.
- [36] T. White, B. Pagurek, and F. Oppacher. ASGA: Improving the ant system by integration with genetic algorithms. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 610–617, Madison, Wisconsin, 1998. Morgan Kaufmann.
- [37] M. Wilson, C. Melhuish, and A. Sendova-Franks. Creating annular structures inspired by ant colony behaviour using minimalist robots. *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, 2:53–58, Oct 2002.
- [38] L. Zadeh. Fuzzy sets. *Information and Control*, pages 338–353, 1965.

APPENDICES

Appendix A Datasets

A.1 Parameters Used to Generate the Gaussian Datasets

Table A.1. Parameters Used to Generate the Gauss Datasets

Dataset	Mean		Std. Dev	
Gauss-1	2	2	1	1
	8	8	1	1
	2	16	1	1
	19	3	1	1
	14	14	1	1
Gauss-2	2	2	1	1
	10	10	1	1
	3	20	1	1
	22	5	2	2
	18	18	1	1
Gauss-3	2	4	1	1
	11	11	1	1
	5	20	1	1
	22	5	2	2
	20	20	1	1
Gauss-4	3	3	1	1
	11	11	1	1
	10	30	1	1
	32	5	1	1
	20	33	1	1
Gauss-5	2	2	1	1
	6	6	1	1
	2	10	1	1
	12	4	2	2
	11	11	1	1

Appendix A (Continued)

Table A.2. Parameters Used to Generate the Gauss-500 Datasets

Dataset	Mean		Std. Dev	
Gauss-500-1	4	4	1	1
	8	8	1	1
	3	12	1	1
	11	4	2	2
	12	12	1	1
Gauss-500-2	4	4	1	1
	7	10	1	1
	4	16	1	1
	13	4	1	1
	14	14	1	1
Gauss-500-3	4	6	1	1
	12	15	1	1
	4	21	1	1
	13	4	1	1
	18	14	1	1
Gauss-500-4	4	6	1	1
	12	16	1	1
	4	21	1	1
	13	4	2	2
	24	14	1	1
Gauss-500-5	6	8	1	1
	15	15	1	1
	9	21	1	1
	16	4	2	2
	28	18	1	1

Appendix A (Continued)

A.2 Plots of the Different Datasets

Figure A.1. Wine Dataset (Normalized)- First 2 Principal Components

Figure A.2. Wine Dataset (Normalized)- First 3 Principal Components

Appendix A (Continued)

Figure A.3. Glass Dataset (Normalized)- First 2 Principal Components

Figure A.4. Glass Dataset (Normalized)- First 3 Principal Components

Appendix A (Continued)

Figure A.5. Glass Dataset: 2 classes (Normalized)- First 2 Principal Components

Figure A.6. Glass Dataset: 2 classes (Normalized)- First 3 Principal Components

Appendix A (Continued)

Figure A.7. Multiple Sclerosis Dataset (Normalized)- First 2 Principal Components

Figure A.8. Multiple Sclerosis Dataset (Normalized)- First 3 Principal Components

Appendix A (Continued)

Figure A.9. Gauss-1 Dataset (Normalized)

Figure A.10. Gauss-2 Dataset (Normalized)

Appendix A (Continued)

Figure A.11. Gauss-3 Dataset (Normalized)

Figure A.12. Gauss-4 Dataset (Normalized)

Appendix A (Continued)

Figure A.13. Gauss-5 Dataset (Normalized)

Figure A.14. Gauss500-1 Dataset (Normalized)

Appendix A (Continued)

Figure A.15. Gauss500-2 Dataset (Normalized)

Figure A.16. Gauss500-3 Dataset (Normalized)

Appendix A (Continued)

Figure A.17. Gauss500-4 Dataset (Normalized)

Figure A.18. Gauss500-5 Dataset (Normalized)