A FUZZY INCREMENTAL CLUSTERING APPROACH TO HYBRID DATA DISCOVERY

Radu D. GĂCEANU, Horia F. POP
Computer Science Department, Babeş-Bolyai University, Cluj-Napoca, Romania,
e-mail: rgaceanu@cs.ubbcluj.ro, hfpop@cs.ubbcluj.ro

ABSTRACT

We propose an incremental fuzzy clustering algorithm for hybrid data discovery. The algorithm is based on the ASM model where data items are represented by agents placed in a two dimensional grid. The agents will group themselves into clusters by making simple moves in their environment. They will try to get closer to each other if they are rather similar or to get away from each other if they are rather different. The algorithm allocates a new agent on the grid whenever a new data item arrives. At each step the new agent contacts an agent from the grid and if they are similar then they will group together in the same cluster. Whenever a new cluster is created the agents will try to merge the cluster with one of the previously created clusters. If a newly created agent does not find a similar fellow then it will start an ASM-like process in order to search for one and thus the data is clustered.

Keywords: *incremental clustering, fuzzy, agents, hybrid data*

1. INTRODUCTION

Several clustering algorithms exist each with its own strengths and weaknesses. Some algorithms need an initial estimation of the number of clusters (k-means, fuzzy c-means); others could often be too slow (agglomerative hierarchical clustering algorithms). Ant-based clustering algorithms often require hybridization with a classical clustering algorithm such as k-means.

In [2] an ant-based clustering algorithm is presented. It is based on the ASM (Ants Sleeping Model) approach. In ASM, an ant has two states on a two-dimensional grid: active state and sleeping state. When the artificial ant's fitness is low, it has a higher probability to wake up and stay in active state. It will thus leave its original position to search for a more secure and comfortable position to sleep. When an ant locates a comfortable and secure position, it has a higher probability to sleep unless the surrounding environment becomes less hospitable and activates it again.

In [3] a Stigmergic Agent System (SAS) combining the strengths of Ant Colony Systems and Multi-Agent Systems concepts is proposed. The agents from the SAS are using both direct and indirect communication. By using direct communication the risk of getting trapped in local optima is lower. However, as showed in [16], most ant-based algorithms can be used only in a first phase of the clustering process because of the high number of clusters that are usually produced. In a second phase a k-means-like algorithm is often used.

In [16], an algorithm in which the behaviour of the artificial ants is governed by fuzzy IF-THEN rules is presented. Like all ant-based clustering algorithms, no initial partitioning of the data is needed, nor should the number of clusters be known in advance. The ants are capable to make their own decisions about picking up items. Hence the two phases of the classical ant-based clustering algorithm are merged into one, and k-means becomes superfluous.

In [6], the clustering problem is approached by the idea of context-aware ASM agents. The agents are able to detect changes in the environment and adjust their moves accordingly. The advantage of this approach is that it enables the ants to communicate directly like in [3] therefore breaking the neighbourhood boundaries and thus decreasing the chance of ants to get trapped in local minima. The fuzzy IF-THEN rules governing the agents' movements are also allowing the agents to go beyond the neighbourhood limits; for example in the case of two very different (VD) agents it makes no point to keep them in a reachable distance; and it makes sense that two very different (VD) agents should move further away from each other than two different (D) agents do. Thus the system behaves more naturally. The agents are able to adapt their movements if changes in the environment occurred and this is an important feature in real-time systems, wireless sensor networks or data streams.

In order to solve the clustering problem we propose an incremental algorithm based on ASM (Ants Sleeping Model) [2, 6]. Incremental clustering is used to process sequential, continuous data flows or data streams and in situations in which cluster shapes change over time. They are well fitted in real-time systems, wireless sensor networks or data streams because in such systems it is difficult to store the datasets in memory. Incremental clustering algorithms in general do not rely on the in-memory dataset and therefore the space and also time requirements for such algorithms are small.

The rest of the paper is structured as follows. In Section 2 the motivation of our approach is outlined followed by the related work in Section 3. In Section 4 the theoretical background is presented. The proposed model is described in Section 5 and Section 6 contains a case study. The advantages and drawbacks of the approach together with some concluding remarks are presented in the closing Section 7.

2. MOTIVATION

Clustering is perhaps the most important unsupervised learning problem and it is widely used in the context of data mining. Unfortunately most of the algorithms are designed for static data, i.e., data that does not change significantly over time. Such algorithms can handle neither changes within existing data items nor newly arrived data items. In order to deal with changes in data these batch algorithms need to recluster the whole dataset which is sometimes infeasible or even impossible. On the other hand incremental clustering algorithms do not rely on the in memory representation of the datasets and hence they can deal with large amounts of data. Moreover if the requirement is to process continuous data flows or if the cluster shapes change over time then incremental clustering will be the best choice. In real life situations cluster boundaries are often not sharp, i.e., clusters are overlapping so fuzzy clustering is more suited in these cases. In fuzzy clustering each data item belongs to a cluster in a certain degree and one data item may belong to more than one cluster indicating the strength of association between data items and clusters. This leads to the discovery of hybrid data items. The presence of hybrid items may be an indication of the quality of data as seen in our experiments from Section 6. But hybrid items may also signify fraudulent behaviour in the financial or banking sector or may indicate an intruder in intrusion detection systems.

Incremental methods are quite a novel topic in cluster analysis research. They are essentially different from online and off-line learning methods. With off-line learning the whole data set is assumed available at all times, and with on-line learning the learning procedure becomes iterative and considers one data item at a time in repetitive turns. On the contrary, incremental learning procedures assume that at each time step the decision is based on updating the data structures based on the data structures constructed at the previous time step. This approach should add an increase in processing speed and robustness as compared with traditional learning methods.

There are a large number of examples suggesting that incremental learning and reasoning are some of the intelligent methods most used by humans in their real life. Such an example is speech recognition, where the listener recognizes and understands the speech of the speaker in incremental steps, before actually having the whole statement available.

As well, when incrementally clustering, humans have the ability to dynamically recognize that the extra data item considered actually contributes to a local reorganization of the data clusters, leading to, for instance, an increase or decrease in the total number of clusters.

The contributions of this paper are as follows: an incremental fuzzy clustering algorithm, the discovery and analysis of hybrid data items, experimental evaluation of the proposed approach on standard datasets.

3. RELATED WORK

In [6], the clustering problem is approached by the idea of context-aware ASM agents. The agents are able to detect changes in the environment and adjust their moves accordingly. Unlike the agents from the classical ASM model [2] the agents from [6] are able to communicate directly therefore breaking the neighbourhood boundaries and thus decreasing the chance of ants to get trapped in local minima. The agents are able to adapt their movements in case of any changes in the environment. However only changes in the features of already existing data items are handled. So clustering new incoming data items is not an issue in the approach from [6]. In this sense the incremental approach from this paper represents a natural step forward.

In [8] an interesting incremental clustering approach is presented similar to the incremental DBSCAN algorithm [5]. In the incremental version of the DBSCAN algorithm data can be added to existing clusters, one point at a time. The main difference of the approach from [8] is that it adds groups of points to existing clusters. The data points to be added are first clustered using the DBSCAN algorithm [4] and the resulted clusters are merged with already existing ones. In other words, instead of adding data items incrementally the algorithm is adding clusters incrementally.

Kamble uses in [9] a genetic algorithm for incrementally cluster data. The genetic algorithm [7] uses and manipulates a population of potential solutions to find the optimal solutions and a generation is completed after each individual in the population has performed the genetic operators. The individuals in the population will be better adapted to the objective or fitness function, as they have to survive in the subsequent generations. The algorithm from [9] works in metric spaces and it is a density-based approach as the key idea is that for each element of a cluster the number of items in the neighbourhood need to be above a certain threshold.

Lee et al. present in [11] a method to implicitly resolve ambiguities using an incremental clustering in Korean to English cross language information retrieval. In their approach a query in Korean is first translated into English using a Korean-English dictionary and then documents are retrieved for the translated query terms. Query-oriented document clusters are incrementally created for the top ranked retrieved documents and the weight of each document retrieved is recomputed based on the clusters created. Considering their experiments the authors conclude that their method outperforms the monolingual retrieval. In [1] an incremental clustering model is considered where the goal is to efficiently attempt to maintain clusters of small diameter as new items to be clustered are arriving.

In [12] an incremental clustering for trajectories is presented. Due to their sequential nature, trajectory data (or moving objects) are often received incrementally as flows of data from various possible sources like GPS. Most of the existing trajectory clustering algorithms are developed for static datasets, but, as the authors remark, such static approach is not suitable when frequent reclustering is needed and when huge amounts of trajectory data are accumulated constantly and needs immediate processing.

In [13] a method for training set compression by using incremental clustering is proposed. The size of the training set can greatly influence the performance of a classifier because it is difficult to be stored in memory and to process it. So reducing the size of a training set is undoubtedly beneficial for a classification process.

4. THEORETICAL BACKGROUND

In machine learning, clustering is an example of unsupervised learning because it does not rely on predefined classes and class-labelled training examples. So it could be said that clustering is a form of learning by observation, rather than learning by examples. In data analysis, efforts have been conducted on finding methods for efficient and effective cluster analysis in large databases. The main requirements for a good clustering algorithm would be the scalability of the method, its effectiveness for clustering complex shapes and types of data, dealing with high-dimensional data, and handling mixed numerical and categorical data in large databases.

Fuzzy logic can be viewed as an extension of the Boolean logic. It's important to note that basically any theory could be fuzzified by replacing the classical sets with fuzzy sets. Fuzzy sets could be applied in modelling inexact behaviour, which was not very convenient via the classical set theory. While the classical set theory deals with well defined objects, the fuzzy set theory deals with objects which have a certain membership degree. There are many applications of fuzzy logic, but actually "is there need for fuzzy logic"? This is the question that Zadeh addresses in [17]. As it is very well explained, fuzzy logic is not fuzzy, it is actually a precise logic of imprecision and approximate reasoning. It is concluded that the progress from bivalent to fuzzy logic is a natural step forward and an important evolution of science.

An agent is an entity that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors [14, 15]. According to [14, 15] agents exhibit the following characteristics: autonomy, reactivity, pro-activity, sociability, intelligence, mobility, self-organization. Usually agents coexist and interact forming Multi-agent Systems (MAS). In computer science, a MAS is a system composed of several interacting agents, collectively capable of reaching goals that are difficult to achieve by an individual agent or monolithic system.

The ACO (Ant Colony Optimization) metaheuristic is composed of different algorithms in which several cooperative agent populations try to simulate real ants behaviour. In ASM (Ants Sleeping Model), an ant has two states on a two-dimensional grid: active state and sleeping state. When the artificial ant's fitness is low, it has a higher probability to wake up and stay in active state. When an ant is in active state it is searching its local neighbourhood for finding better positions in which they may go to sleep again. Since each individual ant uses only a little local information to decide whether to be in active state or sleeping state, the whole ant group dynamically self-organizes into distinctive, clusters.

5. INCREMENTAL FUZZY CLUSTERING

The idea behind incremental clustering is that it is possible to consider one instance at a time and assign it to existing clusters without significantly affecting the already existing structures. Only the cluster representations need to be kept in memory not the entire dataset and thus the space requirements for such an algorithm are very small.

Whenever a new instance is considered an incremental clustering algorithm would basically try to assign it to one of the already exiting clusters. Such a process is not very complex and therefore the time requirements for an incremental clustering algorithm are also small.

5.1. Formal aspects

Our incremental clustering approach is based on the ASM-like algorithm from [2]. In the ASM model, due to the need for security, the ants are constantly choosing a more comfortable environment to sleep in. The ants feel comfortable among individuals having similar characteristics. In ASM, each data item is represented by an agent, and his purpose is to search for a comfortable position for sleeping in his surrounding environment. While he doesn't find a suitable position to have a rest, he will actively move around to search for it and stop when he finds one; when he is not satisfied with his current position, he becomes active again. The definitions 5.1 – 5.6 are taken from [2].

Definition 5.1. *(Chen et al., [2]). The grid in ASM is a two-dimensional array* $G(x,y) \in Z^{+} \bigcup \{0\}$, *of all positions* $(x,y) \in [0..2\lceil\sqrt{n}\rceil - 1]^2$, *such that:*

$$G(x,y) = \begin{cases} i & \text{if there is an agent } i \text{ at position } (x,y) \\ 0 & \text{otherwise} \end{cases}$$

where n is the number of agents.

Remark 5.1. *The size of the grid depends on the number of agents in order to avoid overcrowding the grid or, conversely, allocating a grid which consumes too many resources.*

Remark 5.2. *The ASM uses a grid topologically equivalent to a sphere grid. The advantages of this grid are, on the one hand, it can ensure the equality of all the locations in the grid. So the cells from the margins have the same number of neighbours as the cells from the interior of the grid i.e. one can jump from one of the northmost positions to one of the southmost positions with one step.*

Definition 5.2. *(Chen et al., [2]). In ASM, each agent represents one data object. Let an agent represent a data object by using* $agent_i$ *to represent the* i^{th} *agent, and n be the number of agents. The position of an agent is represented by* (x_i, y_i), *namely* $G(agent_i) = G(x_i, y_i) = i$

Definition 5.3. *(Chen et al., [2]). The neighbourhood of an agent is* $N(agent_i) = N(x_i, y_i) =$
$= \{(x,y) mod(2\lceil\sqrt{n}\rceil) \mid \mid x - x_i \mid \leq s_x, \mid y - y_i \mid \leq s_y\}$
where s_x *and* s_y *are the vision limits in the horizontal and vertical direction respectively.*

Definition 5.4. *(Chen et al., [2]). The set of empty positions in the neighbourhood is*
$L(agent_i) = L(x_i, y_i) =$
$= \{(x,y) | (x,y) \in N(agent_i), G(x,y) = 0\}.$

Definition 5.5. *(Chen et al., [2]). The fitness of an agent is*
$f(agent_i) = \frac{1}{(2s_x+1)(2s_y+1)} \sum_{agent_j \in N(agent_i)} \frac{\alpha^2}{\alpha^2 + d(agent_i, agent_j)^2}$,
$\alpha = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} d(agent_i, agent_j)$,
where:

- $f(agent_i)$ *represents the current fitness of agent i,*
- α *is the average distance between the agents.*

Remark 5.3. *The distance between two agents,* $d(agent_i, agent_j)$, *is the Euclidean distance between the two agents from the grid.*

Definition 5.6. *(Chen et al., [2]). The activation probability is*

$$p_a(agent_i) = cos^{\lambda}(\frac{\pi}{2} f(agent_i)), \tag{1}$$

where $\lambda \in R^+$ is a parameter, and can be called agents' activation pressure.

Remark 5.4. *Function $p_a(agent_i)$ represents the probability of the activation of the agent by the surroundings. If the fitness is low then the probability of activation will be high so the agent is probably going to wake up, move and search for a better place to sleep. Conversely if the fitness is high then the probability will be low so most probably the agent will stay and sleep.*

Definition 5.7. *In our approach we define the fitness in the following way*
$f(agent_i) = \frac{1}{v} \sum_{a_j \in N(a_i)} \frac{\alpha^2}{\alpha^2 + disim(a_i,a_j) de(a_i,a_j)}$,
where:

- *a_i and a_j respectively denote $agent_i$ and $agent_j$,*
- *$de(a_i,a_j)$ represents the Euclidean distance between the agents,*
- *$disim(a_i,a_j)$ denotes the dissimilarity between the two agents,*
- *$v = \frac{1}{(2s_x+1)(2s_y+1)}$.*

5.2. Our approach

As in the algorithm from [2], the agents are randomly scattered on the grid (see Definition 5.1) in active state at the beginning. They randomly move on the grid. In each loop, after the agent moves to a new position, it will recalculate its current fitness f and probability p_a so as to decide whether it needs to continue moving.

If the current p_a is small, the agent will have a lower probability of continuing moving and higher probability of taking a rest at its current position. Otherwise the agent will stay in active state and continue moving. The agent's fitness is related to its similarity with other agents in its neighbourhood.

With increasing number of iterations, such movements gradually increase, eventually, making similar agents gathered within a small area and different types of agents located in separated areas. Thus, the corresponding data items are clustered.

A high level pseudo-code of our incremental clustering algorithm is presented bellow.

```
Algorithm Clustering is
    initialize parameters $\alpha, \lambda, t, s_x, s_y$
    initialize an agent $a_0$ for the first arrived item and
        create a cluster $c_0$ containing agent $a_0$
    while (condition)
        for each data item i
            create an agent $a_i$ (see Definition 5.2)
            randomly place agent $a_i$ on the grid
            j ← random{0, i}
            if $isSimilar(a_i, a_j)$
            then
                $groupSimilar(a_i, a_j)$
            else
                $add(U, a_i)$ // U — the set of unclustered agents
            endif
            if $hasElements(U)$
            then
                $a_{asm} \leftarrow getRandomAgent(U)$
                $tryActivate(a_{asm})$
            endif
        endfor
        adaptively update parameters
    endwhile
endAlgorithm
```

The algorithm continuously receives new data items to be clustered. For the first such item, a new agent is created that encapsulates this item. As in the classical ASM model, one agent per item will be allocated (see Definition 5.2).

A new cluster is created (the first one) and the agent is added to this cluster. In the following, whenever a new item i arrives, a new agent, a_i, is created. A random position from the grid is selected in order for the agent to be be placed on; the process continues until an empty position is found.

Afterwards the agent contacts an already existing agent from the grid. If they are similar then they are grouped together otherwise the agent a_i is added to the set of unclustered agents, U. In order to compute the similarity between agents the fuzzy variable called *Similarity* from Figure 1 is used; in Figure 1 the real values denoting similarities are mapped to states of the considered fuzzy variable. We use the Euclidean distance as a dissimilarity measure in our experiments. The $groupSimilar(a_i,a_j)$ procedure groups together two agents, i.e., the agent a_i moves towards the agent a_j on the grid (i.e. moves as close as possible to a_j on an empty position from his neighbourhood) and also it is added to the a_j's cluster. If a_j is not already in a cluster then a new cluster containing both agents will be created.

Whenever a new cluster is created we try to merge it with an already existing cluster. The merging is performed based on the similarity between the cluster representatives. The representative of a cluster is either the first item that was added to the cluster or an item pointed out, i.e., manually chosen by the data analyst. The similarity between two agents will be discussed immediately. To the cluster representatives we will come back in the experiments section.

In the final step of the for loop we test if the set of unclustered agents is not empty and if it contains elements then we will try to activate a randomly extracted agent from there. In $tryActivate(a_{asm})$ the agent will follow an ASM-like clustering process similar to the one from [6]. The basic idea is that the agents move while they are in active state (see Definition 5.6 — probability of activation). The agents select an empty position from the set L (see Definition 5.4) to move to. The size of the considered neighbourhood depends on the dissimilarity level (D or VD). In our case $s_x = s_y = 2$ for D and $s_x = s_y = 4$ for VD. If an agent finds a similar fellow then it will call the $groupSimilar(a_i,a_j)$ procedure. The whole process ends when the set of unclustered items contains an acceptable number of elements.

In [6], the agents decide upon the way they move on the grid according to their similarity with the neighbours, using fuzzy IF-THEN rules. Thus two agents can be similar (S), different (D), very different (VD). The fuzzy sets S, D, VD are defined as follows:

$$S,D,VD : X \to [0,1] \tag{2}$$

$$S(x) = \begin{cases} 1 & ,x \in [0,SD1] \\ (SD1-x)+1 & ,x \in [SD1,SD2] \\ 0 & ,otherwise \end{cases} \tag{3}$$

$$D(x) = \begin{cases} (x-SD2)+1 & ,x \in [SD1,SD2] \\ 1 & ,x \in [SD2,VD1] \\ (VD1-x)+1 & ,x \in [VD1,VD2] \\ 0 & ,otherwise \end{cases} \tag{4}$$

$$VD(x) = \begin{cases} (x-VD2)+1 & ,x \in [VD1,VD2] \\ 1 & ,x > VD2 \\ 0 & ,otherwise \end{cases} \tag{5}$$

If two agents are similar or very similar they will get closer to each other. If they are different or very different they will get away from each other. The number of steps they do each time they move depend on the similarity level. So if the agents are VD they will jump many steps away from each other; if they are D they will jump less steps away from each other. In the end the ants which are S will be in the same cluster. The similarity computation is taking into account the actual structure of the data or the data density from the agent's neighbourhood; a bigger change from one agent to another translates into a certain similarity which then affects the agent's movement on the grid. A graphical representation of a fuzzy variable $Similarity$ is shown in Figure 1.

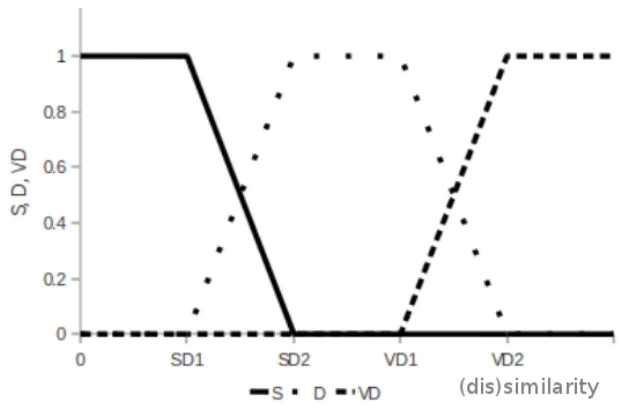

Fig. 1 The fuzzy sets S, D and VD corresponding respectively to the linguistic concepts $Similar$, $Different$ and $VeryDifferent$ are called the $states$ of the fuzzy variable $Similarity$. The limits $SD1$, $SD2$, $VD1$, $VD2$ are application specific

The parameter α is the average distance between agents and this changes at each step further influencing the fitness function. The parameter λ influences the agents' activation pressure and it may decrease over time. The parameter t (the number of iterations) is used for the termination condition which could be something like $t < t_{max}$. In the considered experiments we allow only one execution of the $while$ loop, so we initialize $t = 0$ and we set $t_{max} = 1$. The parameters s_x, s_y, the agent's vision limits may also be updated in some situations.

6. EXPERIMENTS

In order to test the algorithm in a real-world scenario, the Iris dataset [18] was considered for a first test case. The data set contains 3 classes of 50 instances each, each class referring to a type of iris plant. There are 4 attributes plus the class: sepal length in cm, sepal width in cm, petal length in cm, petal width in cm, class (Iris Setosa, Iris Versicolour, Iris Virginica). This dataset is appropriate for rather testing classification, but it was preferred for clustering too because the class attribute is given and hence there is a way to evaluate the algorithm. So apparently it would be ideal for the algorithm to produce 3 clusters of 50 instances, the 3 clusters corresponding to the given 3 classes.

The last 2 attributes (petal length in cm and petal width in cm) are highly correlated according to [18]. However we do not dismiss any of these attributes because we would like to keep as much of the data unchanged. We do however scale the data to the interval $[0,1]$. At this point the clustering process can be started. In the final grid configuration one would clearly see the clustered agents. Due to space limitations we can't show here the final grid configuration, but we will immediately summarize the results. Besides the final grid configuration a membership table is also produced. The membership table shows the membership degree of each agent to the clusters. Relevant parts of the membership table will also be explained immediately.

According to the Iris dataset [18], items ranging from 0 to 49 belong to the first class, items ranging from 50 to 99 belong to the second class and items ranging from 100 to 149 belong to the third class. So from the grid table it appears that the following clusters contain some misclassifications:

- $Cluster1$ (items 0 – 49): no misclassifications
- $Cluster2$ (items 50 – 99): 106, 119, 133, 134
- $Cluster3$ (items 100 – 149): 70, 77

So it appears that the algorithm has misclassified 6 items.

Let us check each item in more detail. In order to do this deeper analysis we have to check out the membership degrees table. In the membership table one can see in what degree does each item belong to the clusters. This membership is computed with respect to the cluster representatives. The representative of a cluster is either the first item from that cluster or an item designated by the data analyst. In case of cluster merging if any of the representatives is designated by data analyst then this will be the representative of the new cluster; otherwise the representative of the cluster with the greatest number of elements will be chosen as the new representative. The agents 45, 95, 145 resulted from this process as final representatives for clusters 1, 2 and 3 respectively.

In the following, the similarity between each of the above reported misclassification and the corresponding representative is given. The membership degree to each cluster is also considered. The Euclidean distance between items has been used as a similarity metric. So a small similarity value i.e. distance between two items means that the two items are similar, should stay together.

Table 1 Cluster2 — RepresentativeId (95)

MisclassificationId	Similarity	C1	C2	C3
106	0.25	0	0.95	0.84
119	0.24	0	0.96	0.83
133	0.19	0	1	0.89
134	0.24	0	0.96	0.82

From the Table 1 it can be seen that the similarities between the considered items and the representative lie between 0.19 and 0.25. This makes them S ($Similar$) with this item. The membership degree with $Cluster2$ suggests that these items belong to this cluster. However the membership degree with $Cluster3$ is also high. The highest membership degree is with $Cluster2$ though and because of this it could be claimed that the items are actually correctly classified with respect to the considered metric.

However we believe that these items cannot be considered to strictly belong either to $Cluster2$ or to $Cluster3$ as they are clearly at the border of the two clusters so they belong to both. In this case we also believe that they should not be regarded as misclassifications.

Let us check the reported misclassifications from the third cluster.

Table 2 Cluster3 — RepresentativeId (145)

MisclassificationId	Similarity	C1	C2	C3
70	0.22	0.0	0.94	0.98
77	0.24	0.0	0.95	0.96

Like the items from Table 1, the items from the Table 2 should also not count as misclassifications for the same reasons from above. So, arguably, the algorithm has a 100% classification accuracy for the considered dataset.

It may be observed that starting with item 50 a lot of instances have high membership degrees to both $Cluster2$ and $Cluster3$. This suggests that the members of these two classes are not linearly separable as mentioned in [18] so a clustering process could also merge these two clusters and hence report only two clusters instead of three.

For the second case study the wine dataset [19] was considered. This dataset contains the results of a chemical analysis of wines grown in the same region in Italy but derived from three different wine growers. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

In [19] it is mentioned that the initial dataset had 30 attributes. So the current dataset has 13 attributes plus the class. There are 178 instances grouped in three classes corresponding to the three wine growers. Items ranging from 1 to 59 belong to the first class, items from 60 to 130 belong to the second class and items from 131 to 178 belong to the third class.

The same procedure as the one for the first case study was applied and the following misclassifications resulted from the final grid configuration:

- $Cluster1$ (items 0 – 58): 63, 65, 66, 73, 78, 95, 98
- $Cluster2$ (items 59 – 129): 130, 134
- $Cluster3$ (items 130 – 177): 83

From the above it appears that the algorithm has misclassified 10 items. Moreover, the following items have not been classified, i.e., they remained in the collection U of unclustered agents: 59, 110, 121, 123, 124. So it appears that there are 15 classification errors. But let us check out everything in more detail.

The agents 17, 89, 166 are the final representatives for clusters 1, 2 and 3 respectively.

In the following, the similarity between each of the above reported misclassification and the corresponding representative is given. The membership degree to each cluster is also considered. The Euclidean distance between items has been used as a similarity metric. So a small similarity (less than 0.7) value i.e. distance between two items means that the two items are similar, should stay together.

Table 3 Cluster1 — RepresentativeId (17)

MisclassificationId	Similarity	C1	C2	C3
63	0.63	0.87	0.83	0
65	0.42	1	0.99	0
66	0.64	0.86	0.83	0
73	0.61	0.89	0	0
78	0.69	0.81	0	0
95	0.69	0.81	0	0
98	0.51	0.99	0.8	0

Table 4 Cluster2 — RepresentativeId (89)

MisclassificationId	Similarity	C1	C2	C3
130	0.76	0	0	0
134	0.68	0	0.82	0.8

Table 5 Cluster3 — RepresentativeId (166)

MisclassificationId	Similarity	C1	C2	C3
83	0.6	0	0.81	0.9

Table 6 Unclustered items

ItemId	SimC1	C1	Sim C2	C2	Sim C3	C3
59	1.06	0	0.75	0	1.08	0
110	0.91	0	0.93	0	1.11	0
121	0.73	0	0.96	0	1.2	0
123	1	0	0.9	0	0.99	0
124	0.94	0	0.86	0	1.13	0

In Table 3 it can be seen that the similarities between the considered items and the representative are bellow 0.7. This makes them $S(Similar)$ with this item. The membership degree with $Cluster1$ suggests that these items belong to this cluster. However the membership degree with $Cluster2$ is also high. The highest membership degree is with $Cluster1$ though and because of this it could be claimed that the items are actually correctly classified with respect to the considered metric. Items 73, 78 and 95 clearly belong to $Cluster1$ since they have a 0 membership degree with respect to $Cluster2$ according to the considered metric; indeed, their dissimilarities with respect to the $Cluster2$ representative (item 89) are 0.77, 0.71, 0.86 respectively. Since $SD1 = 0.5$ and $SD2 = 0.7$, from Relation 3 it follows that the membership degree with $Cluster2$ is 0.

As it can be seen from Table 6, all the elements from the collection U have very high similarity values with respect to each cluster and this implies membership degrees equal to zero. So these items are correctly left apart of any cluster.

Consequently, it can be argued that only item 130 is truly a misclassification with respect to the considered metric. On the other hand, it is clear that the approach has 15 classification errors if the results from [19] were to be taken ad litteram. This may suggest that another metric should be considered for computing the similarity between items. From a classification point of view, even in the most pessimistic result interpretation (15 classification errors), the accuracy would be 91%.

Bottom line: we have seen in both tests that most of the apparent classification errors were actually items that have high membership degrees to more than one cluster. Nevertheless, in our opinion, it is clear that we are dealing with hybrid data. Actually the hybrid nature of the data is suggested in [18] where it is stated that one class is linearly separable from the other two, but the latter are not linearly separable from each other. By using fuzzy methods such features of the data are easy to be observed. We have also discovered hybrid data in the case of the wine dataset [19]. In this case we assume that the quality of data is not good enough in order for clearly separate between different wine breeds. This idea is strengthened by [19] where it is stated that the dataset initially had around 30 variables, but the current one only has 13 variables. So it is unimportant if in our experiments we have obtained a certain number of classification errors with respect to the results from [18, 19] as long as our approach allows us to have an insight into these data such that we can actually see that we deal with hybrid items.

7. CONCLUSION

The algorithm we have presented is based on the adaptive ASM approach from [2]. When being in ASM mode, the agent movements are following fuzzy IF-THEN rules for deciding upon the direction and length of the movement like in [6]. The major difference is that the clustering approach is an incremental one. Incremental clustering is used to process sequential, continuous data flows or data streams and in situations in which cluster shapes change over time. They are well fitted in real-time systems, wireless sensor networks or data streams because in such systems it is difficult to store the datasets in memory. The algorithm considers one instance at a time and it basically tries to assign it to one of the existing clusters. Only cluster representations need to be kept in memory so computation is both fast and memory friendly. Experiments on real-world datasets [18, 19] show good clustering results and suggest that the method is a promising one. The fuzziness of the approach allows the discovery of hybrid items in both datasets [18, 19]; the fact that there are hybrid items could be an indication of the quality of data. More experiments with other clustering methods using larger, real-world data sets are on-going.

ACKNOWLEDGEMENT

The authors wish to thank for the financial support provided from programs co-financed by The Sectorial Operational Programme Human Resources Development, Contract POSDRU 6/1.5/S/3 — "Doctoral studies: through science towards society".

REFERENCES

[1] CHARIKAR, M. – CHEKURI, C. – FEDER, T. – MOTWANI, R.: Incremental Clustering and Dynamic Information Retrieval. Proceedings of the twenty-ninth Annual ACM Symposium on Theory of Computing, STOC '97, pp. 626–635, ACM.

[2] CHEN, L. – XU, X. H. – CHEN, Y. X.: An Adaptive Ant Colony Clustering Algorithm. Proceedings of International Conference on Machine Learning and Cybernetics 2004, pp. 1387–1392.

[3] CHIRA, C. – DUMITRESCU, D. – GĂCEANU, R. D.: Stigmergic Agent Systems for Solving NP-hard problems. Studia Informatica, Special Issue, KEPT-2007: Knowledge Engineering: Principles and Techniques, June 2007, pp. 177–184.

[4] ESTER, M. – KRIEGEL, H.-P. – JÖRG, S. – XIAOWEI, X.: A Density-based Algorithm for Discov-

ering Clusters in Large Sdatabases with Noise. AAAI Press, 1996, pp. 226–231.

[5] ESTER, M. – KRIEGEL, H.-P. – JÖRG, S. – WIMMER, M. – XIAOWEI, X.: Incremental Clustering for Mining in a Data Warehousing Environment. Proceedings of the 24rd International Conference on Very Large Data Bases, VLDB ’98, 1998, pp. 323–333, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA.

[6] GĂCEANU, R. D. – POP, H. F.: A Context-Aware ASM-Based Clustering Algorithm. Studia Universitatis Babes-Bolyai Series Informatica, Vol. LVI, No. 2, 2011, pp. 55–61.

[7] GOLDBERG, D. E.: Genetic Algorithms in Search. Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., Inc., 1989.

[8] GOYAL, N. – GOYAL, P. – VENKATRAMAIAH, K. – DEEPAK, P. C. – SANNOP, P. S.: An Efficient Density Based Incremental Clustering Algorithm in Data Warehousing Environment. 2009 International Conference on Computer Engineering and Applications, IPCSIT, Vol. 2, 2011, IACSIT Press, Singapore.

[9] KAMBLE, A.: Incremental Clustering in Data Mining using Genetic Algorithm. International Journal of Computer Theory and Engineering, Vol. 2, No. 3, June 2010.

[10] HARTIGAN, J. A.: Clustering Algorithms, Wiley series in probability and mathematical statistics. Applied probability and statistics, 1975.

[11] LEE, K. S. – KAGEARA, K. – CHOI, K. S.: Implicit Ambiguity Resolution using Incremental Clustering in Korean-to-English Cross-Language Information Retrieval. Proceedings of the 19th International Conference on Computational Linguistics, Vol. 1, COLING ’02, 2002, pp. 1–7, Association for Computational Linguistics, Stroudsburg, PA, USA.

[12] LI,Z. – LEE, J. G. – LI, X. – HAN, J.: Incremental Clustering for Trajectories. Database Systems for Advanced Applications, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, pp. 32–46, 2010.

[13] LI, D. – SIMSKE, S.: Training Set Compression by Incremental Clustering. Journal of Pattern Recognition Research, Vol. 6, No. 1, 2011, pp. 56–64.

[14] SERBAN, G.: Sisteme Mutiagent in Inteligenta Artificiala Distribuita. Arhitecturi si aplicatii, Cluj-Napoca: Ed. Risoprint, 2006.

[15] SERBAN, G. – POP, H. F.: Tehnici de Inteligenta Artificiala. Abordari bazate pe Agenti Inteligenti, Cluj-Napoca: Ed. Mediamira, 2004.

[16] SCHOCKAERT, S. – COCK, M. D. – CORNELIS, C. – KERRE, E. E.: Fuzzy Ant Based Clustering. Ant Colony Optimization and Swarm Intelligence, 4th International Workshop (ANTS 2004), LNCS 3172, 2004, pp. 342–349.

[17] ZADEH, L. A.: Is There a Need for Fuzzy Logic? Information Sciences, Vol. 178, No. 13, July 2008, pp. 2751–2779.

[18] UCI Machine Learning Repository, Iris Data Set, http://archive.ics.uci.edu/ml/datasets/iris

[19] UCI Machine Learning Repository, Wine Data Set, http://archive.ics.uci.edu/ml/datasets/wine

Received January 1, 2012, accepted May 5, 2012

BIOGRAPHIES

Radu D. Găceanu is an Assistant at the Department of Computer Science, Faculty of Mathematics and Computer Science from the Babes-Bolyai University of Cluj-Napoca, Romania. He has an international experience as *Java* and *C#* Software Developer. He is currently involved in a joint Ph.D. program at the Babes-Bolyai University of Cluj-Napoca and the Eötvös Loránd Unviversity of Budapest. His main research interests are Intelligent Data Analysis, Soft Computing, Agent Based Computing.

Horia F. Pop is a Professor at the Department of Computer Science, Faculty of Mathematics and Computer Science from the Babes-Bolyai University of Cluj-Napoca, Romania and Vice-Dean at the aforementioned Faculty. He is an author and co-author of approximately 125 publications out of which 36 are indexed *ISI*. He is also an author and co-author of several books and lecture notes. He has been a member and director of several research grants financed by CNCSIS — Romanian National Higher Education Research Council. His main research interests are Intelligent Data Analysis, Soft Computing, Artificial Intelligence.