Ant based Clustering using case based reasoning

Zahra Sadeghi ¹, Mohammad Teshnehlab ²

¹ Computer Engineering Department,
Science and Research Branch, Islamic Azad University - Member of Young Researchers Club,
Hesarak, Poonak, Tehran, IRAN

² Faculty member, Electronic Engineering Department, K.N.Toosi University,
Seyyed Khandan Bridge, Shariati Street, Tehran, IRAN

Abstract: - In this paper a new ant clustering algorithm based on case based reasoning (CBR) is presented. Every ant has a case base which is updated iteratively by the process of CBR. Each ant can use its case base to find best places for dropping its load. Also ants can take advantage of the knowledge of other ants’ case bases by the process of cooperation. Our simulation results demonstrated better performance in terms of accuracy and compactness of generated clusters than previous approaches of ant based clustering.

Keywords: - Ant based clustering, Ant colony optimization, Case based reasoning

1. Introduction

Clustering is an important technique that has been studied in various fields with many applications such as image processing, marketing, data mining and information retrieval. The goal of clustering is partitioning data into some groups so as all the members of one group have a close relationship to each other. This relationship is often expressed as similarity/dissimilarity measurement and is calculated through distance function. Recently, algorithms inspired by nature are used for clustering. The swarm intelligence clustering models and algorithms have advantages in many aspects, such as no need of priori information, self-organization, flexibility, robustness and decentralization [1]. Ant Colony Optimization (ACO) is a relatively new and expanding branch of intelligent systems. Algorithms based on ant colony are very attractive because they are able to cluster data where there is no prior information about the number and the shape of clusters [2].

In the present article, we have introduced an ant clustering algorithm using case based reasoning. After each successful dropping, the value and the coordination of the object being dropped is saved in the ant’s case base. In the beginning, the case bases of all ants are empty. Whenever an ant picks up a new load, it searches its case base for the best match and if such a case was found, it will be retrieved and then the ant immediately drops down its load in the nearest neighborhood of the grid coordination of the stored case. Otherwise it uses the drop down function. The case base of all ants updates each time they drop down an object. Also ants are able to communicate to each other to find the best place for dropping their load.

2. The general principle of ant clustering

Ant based clustering is a distributed process. The pioneers of this work are Deneubourg et al. [3]. Their model is known as basic model (BM) in which ants are simulated by simple agents that randomly move in an environment which is a square grid. Initially, each data object that represents a multi-dimensional pattern is randomly distributed over the 2-D space. Data items that are scattered within this environment can be picked up, transported and dropped by the agents in a probabilistic way. The picking and dropping operations are influenced by the similarity and density...
of the data items within the ant’s local neighborhood. Deneubourg et al. proposed a computational model for spatial sorting. Their model is based on the following probability of picking and dropping functions:

\[ P_p = \left( \frac{K_p + f}{K_p} \right)^2. \]  
\[ (1) \]

\[ P_d = \left( \frac{f}{K_d + f} \right)^2. \]  
\[ (2) \]

Where \( P_p \) is the probability of picking, \( P_d \) is the probability of dropping, and \( K_p \) and \( K_d \) are constants. \( f \) is the perceived fraction of items in the neighborhood of the ant. The probability of picking \( (P_p) \) is increased if a data object is surrounded by dissimilar data, or when there’s no data in its neighborhood. Also, ants trend to drop data in the vicinity of similar ones, so the probability of dropping \( (P_d) \) is increased if ants are surrounded with similar data.

Deneubourg et al.’s model was later extended by Lumer et al. [4]. In their model which is known as LF model they modified the probability functions as the followings:

\[ P_p(i) = \left( \frac{K_p}{K_p + f(i)} \right)^2. \]  
\[ (3) \]

\[ P_d(i) = \begin{cases} 
2f(i) & \text{if } f(i) < K_d \\
1 & \text{if } f(i) \geq K_d.
\end{cases} \]  
\[ (4) \]

Where \( P_p \) and \( P_d \) are as before, i.e., the probability of dropping and picking. In addition, they introduced the density function as the following:

\[ f(i) = \max\left(0, \frac{1}{\sigma^2} \sum_{j \in L} (1 - \frac{\delta(i, j)}{\alpha}) \right). \]  
\[ (5) \]

Where \( \delta(i, j) \in [0,1] \) is the dissimilarity function between two data object \( i \) and \( j \). \( \alpha \in [0,1] \) is the scaling parameter that permits further adjustments of resulting values. \( \sigma^2 \) is the size of local neighborhood \( L \) around the ant’s current position. The basic ant clustering algorithm is like the one summarized as bellow [2], [5]:

Let \( R \in [0,1] \) be a random number drawn for each use, the basic algorithm is then given by:

Randomly scatter data items on a square grid.

For a number of steps or until a criterion is met, repeat for each ant:

- If the ant is unladen and placed on location \( l \) occupied by object \( o \), the object is picked up if \( R \leq p_{pick\_up}(o) \).
- If the ant is carrying object \( o \) and placed on an empty location \( l \), the object is dropped if \( R \leq p_{drop\_down}(o) \).
- Move the ant to a new randomly selected (neighboring) location.

3. Case Based Reasoning (CBR)

CBR is a familiar process that is a normal, intuitive method of decision-making for humans in everyday life. It is a process of considering past cases and arriving at decisions on comparison between the current situation and the old cases [7].

In CBR terminology, a case usually denotes a problem situation. A previously experienced situation, which has been captured and learned in a way that it can be reused in the solving of future problems, is referred to as a past case, previous case, stored case, or retained case. Correspondingly, a new case or unsolved case is the description of a new problem to be solved [8].

CBR is done by considering the similarity between the current situation and old seen situations. The basic idea of CBR for solving a new problem is as follows:

- Remembering a previous similar situation.
- Comparing the new problem to the old solved problems
- Reusing information and knowledge of that situation.
- Revising the proposed solution
- Retaining the parts of this experience likely to be useful for future problem solving.

At the highest level of generality, a general CBR cycle may be described by the following four processes [8]:

1. RETRIEVE the most similar case or cases
   The goal of this step is to retrieve old cases stored in the case library [9]. Retrieving is the selection of the most similar cases (source cases) to the new problem specifications (target case) from the library of cases.
2. REUSE the information and knowledge in that case to solve the problem
3. REVISE the proposed solution
   Because a new case may not exactly match the old one, the old knowledge may often need to be fixed to fit the new one [9].
4. RETAIN the parts of this experience likely to be useful for future problem solving.
4. The Framework of the Algorithm

Our ant clustering algorithm uses the same probability and density function as the LF model proposed by Lumer et al. [4], but instead of consuming a long time for finding an item to pick, we’ve kept the location of all available items on the grid in an array. After an ant drops its load, it immediately picks up one item from the list of available items on the grid [10].

We have considered a grid with s*s cells. The number of cells must be greater than the number of all objects. All ants move with different speeds on the grid. We have considered a maximum speed for all ants which is set as max_speed=s/3. The step of movement of each ant is generated randomly between 1 and max_speed before each movement. We decrease the max_speed by one every 200 iterations. The details of our proposed method are discussed in the following subsections.

4.1 Case and Case Base of Ants

Each ant has a private case base for keeping the different cases it has seen. A case is consisted of value of attributes and the place they have been dropped. The place of dropping, is a two dimensional matrix (because the search space is a grid). The value of attributes of a data is a multidimensional matrix, in which the number of dimensions depends on the dimensions of data. The size of all case bases are set to 8.

4.2 Retrieve and Revise

Whenever an ant wants to put its load, it searches for a case in its case base, and compares the value of all cases with its current load to find the most similar case. For this, each ant first collects all the cases having a distance less than a threshold from its load in an array, then it sorts the array, and finally selects the first element of the array which has the lowest distance value. But, before retrieving this case, it first checks whether the case is a valid one, or not. Because ants picks up and drop down the data iteratively, it is so probable that the value of case bases become invalid during clustering process. A case is valid for an ant, only if its current location on the grid is the same as the one recorded in the case base of that ant. If the first element of the array was invalid, the second element is checked. This process repeats until a case is retrieved or no other case remains in case base. If a case was retrieved it can be used. It is possible that whenever an ant is looking for a case, it faces a case with identical attribute values with its current load. Under this condition, the case must be revised and then updated. For this purpose, the dropping probability must be computed. When a suitable location for dropping that item is found, the old stored location in case base is replaced with a new one.

4.3 Cooperation between Ants

The information of all ants about the promising cells for dropping objects is different, so ants can take advantage of the information of other ants by the process of cooperation. Whenever a laden ant fails in dropping its load, and an unladen ant fails for picking up an item, they can sense their neighborhood for getting help from other ants. So, a laden ant can put its load on an unladen ant, and an unladen ant can get load from a laden ant. If both ants were laden, they only exchange their loads. Nothing will happen if both were unladen. If there were more than one neighbor ant, one is chosen randomly. This can prevent ants from moving aimlessly and it can also increase the speed of convergence.

4.4 The algorithm of ant clustering using CBR

The details of our algorithm are as follows:

//Initialization part
1. Spread the data and ants randomly on the grid and give all ants a random load.

//main part
1. While iter < max_iter
2. For i=1 to #ants
3. a = Choose a random ant
4. step = Generate a random speed from interval (1, max_speed)
5. new_location = Move (a, step)
6. if ant a is unladen and there is an object o in location l then pick_done = try_pick(o)
7. if pick_done = 0 then try to find a laden ant in the neighborhood of ant a and take the load of it with probability of .5
8. end if
9. end if
10. if ant a is laden then search its case base for a case which has the similarity of more than .9 with the current load, if such a case was found then
11. drop the load in a neighborhood of 2 from the place of that case
12. end if
13. if a case was found with the same value with the current load then
   // revise
14. find a place for dropping it and change the place part of the case with the new found location
15. end if
16. if ant a is laden with load l and no similar case was found and the current location of it is empty
   then drop_done=try_drop(l)
17. end if
18. if drop_done=1
   // retain
19. make a new case and save it in case base
20. end if
21. if drop_done=0
22. find an unladen ant in the neighborhood of 2 and give the load of ant a to it with probability of .5
23. end if
24. End For
25. End while

try_pick(o)
1. r= a random number from interval (o,1)
2. p=compute probability function (3)
3. if r>p then
4. pick up the object o
5. pick_done=1
6. else
7. pick_done=0
8. end if

try_drop(l)
1. r= a random number from interval (o,1)
2. d=compute probability function (4)
3. if r>d then
4. drop down load l
5. drop_done=1
6. else
7. drop_done=0
8. end if

5. Experimental Results and Comparison
The algorithm is applied to four datasets and all except for Iris and Wine dataset are shown visually in "Fig. 1". The results of applying k-means algorithm and ant clustering using LF model are compared to the results of our algorithm in Table 2 and the best result out of a few runs is selected and is illustrated in "Fig. 2". Random initialization is used in our experiments for k-means algorithm and the true numbers of clusters are provided for it. Compared to k-means algorithm and normal ants, our approach shows that it can obtain satisfying results.

Table 1. summary of the used data sets. dim is the dimensionality, k gives the number of clusters, and size shows the number of members of a cluster.

<table>
<thead>
<tr>
<th>Name</th>
<th>k</th>
<th>dim</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square6-4*50</td>
<td>2</td>
<td>N([0,0],[2,2]),([0,6],[2,2]),([6,0],[2,2]),([6,6],[2,2])</td>
<td></td>
</tr>
<tr>
<td>Iris-3</td>
<td>4</td>
<td>UCI</td>
<td></td>
</tr>
<tr>
<td>(50,50,50)</td>
<td>13</td>
<td>UCI</td>
<td></td>
</tr>
<tr>
<td>Wine-3</td>
<td>13</td>
<td>UCI</td>
<td></td>
</tr>
<tr>
<td>(59,71,48)</td>
<td>13</td>
<td>UCI</td>
<td></td>
</tr>
<tr>
<td>D2C4-4</td>
<td>2</td>
<td>Artificial data</td>
<td></td>
</tr>
<tr>
<td>(35,57,84,52)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig 1. Benchmark Data. a) Square6, b) D2C4

Fig 2. Best visual results of benchmark data. a) Square6, b) Iris, c) D2C4, d) Wine
Table 2. Test Results on Benchmark Dara

<table>
<thead>
<tr>
<th></th>
<th>Square6</th>
<th>K-means</th>
<th>LF</th>
<th>Proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Min Error</strong></td>
<td>M</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Max Error</strong></td>
<td>M</td>
<td>10</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td><strong>Average Error</strong></td>
<td>M</td>
<td>2.9</td>
<td>2.8</td>
<td>2.77</td>
</tr>
<tr>
<td><strong>Average Number of Detected Clusters</strong></td>
<td>M</td>
<td>4</td>
<td>4.4</td>
<td>4.05</td>
</tr>
<tr>
<td><strong>Maximum Iterations</strong></td>
<td>M</td>
<td>100</td>
<td>500000</td>
<td>100000</td>
</tr>
<tr>
<td><strong>Iris</strong></td>
<td>K-means</td>
<td>4</td>
<td>3.12</td>
<td>2.91</td>
</tr>
<tr>
<td><strong>Min Error</strong></td>
<td>M</td>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td><strong>Max Error</strong></td>
<td>M</td>
<td>17</td>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td><strong>Average Error</strong></td>
<td>M</td>
<td>8.8</td>
<td>8.95</td>
<td>8.72</td>
</tr>
<tr>
<td><strong>Average Number of Detected Clusters</strong></td>
<td>M</td>
<td>3</td>
<td>3.12</td>
<td>2.91</td>
</tr>
<tr>
<td><strong>Maximum Iterations</strong></td>
<td>M</td>
<td>100</td>
<td>500000</td>
<td>100000</td>
</tr>
<tr>
<td><strong>Wine</strong></td>
<td>K-means</td>
<td>4</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td><strong>Min Error</strong></td>
<td>M</td>
<td>8</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td><strong>Max Error</strong></td>
<td>M</td>
<td>17</td>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td><strong>Average Error</strong></td>
<td>M</td>
<td>8.13</td>
<td>8.21</td>
<td>7.75</td>
</tr>
<tr>
<td><strong>Average Number of Detected Clusters</strong></td>
<td>M</td>
<td>3</td>
<td>3.28</td>
<td>3.12</td>
</tr>
<tr>
<td><strong>Maximum Iterations</strong></td>
<td>M</td>
<td>100</td>
<td>500000</td>
<td>100000</td>
</tr>
<tr>
<td><strong>D2C4</strong></td>
<td>K-means</td>
<td>6</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td><strong>Min Error</strong></td>
<td>M</td>
<td>6.8</td>
<td>5.85</td>
<td>4.17</td>
</tr>
<tr>
<td><strong>Max Error</strong></td>
<td>M</td>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td><strong>Average Error</strong></td>
<td>M</td>
<td>2.7</td>
<td>2.5</td>
<td>2.5</td>
</tr>
<tr>
<td><strong>Average Number of Detected Clusters</strong></td>
<td>M</td>
<td>4</td>
<td>5.85</td>
<td>4.17</td>
</tr>
<tr>
<td><strong>Maximum Iterations</strong></td>
<td>M</td>
<td>100</td>
<td>500000</td>
<td>100000</td>
</tr>
</tbody>
</table>

6. Conclusion

In this article we presented a novel method for ant clustering which uses case based reasoning. We have proposed a method by which ants have the ability of making decision about the best place for dropping their load using their case bases. The case base of each ant contains different information about promising places for dropping loads. We have added a mechanism for taking advantage of the knowledge of other ants. Those ants that were unsuccessful in dropping their loads can give their loads to other unladen ants in their neighborhood, and those that were unsuccessful in picking up an object can take the loads of other laden ants in their neighborhood. If two neighboring ants both were laden they can exchange their loads. This process speeded up the process of clustering and prevented from producing too many small clusters which is one of main drawbacks of ant based clustering.

7. References