Clustering Ensembles Using Ants Algorithm

Javad Azimi, Paul Cull and Xiaoli Fern

EECS Department, Oregon State University,
Corvallis, Oregon, 97330, USA
{azimi,pc,xfern}@EECS.oregonstate.edu

Abstract. Cluster ensembles combine different clustering outputs to obtain a better partition of the data. There are two distinct steps in cluster ensembles, generating a set of initial partitions that are different from one another, and combining the partitions via a consensus functions to generate the final partition. Most of the previous consensus functions require the number of clusters to be specified a priori to obtain a good final partition. In this paper we introduce a new consensus function based on the Ant Colony Algorithms, which can automatically determine the number of clusters and produce highly competitive final clusters. In addition, the proposed method provides a natural way to determine outlier and marginal examples in the data. Experimental results on both synthetic and real-world benchmark data sets are presented to demonstrate the effectiveness of the proposed method in predicting the number of clusters and generating the final partition as well as detecting outlier and marginal examples from data.

Key words: Cluster ensembles, ant colony, consensus function, outlier and marginal samples.

1 Introduction

There is no clustering algorithm that performs best for all data sets. Choosing a single clustering algorithm and appropriate parameters for a data set requires both clustering expertise and insights about the data set itself. Instead of running the risk of picking an inappropriate clustering algorithm, we can leverage the different options available by applying all of them to the data and then combining their clustering results. This is the basic idea behind cluster ensembles [1].

There are two major steps in cluster ensembles: first generating a set of different partitions and then combining them using a consensus function. A major difficulty in cluster ensembles is the design of the consensus function for combining the partitions into a final clustering solution. The combination of multiple partitions can be viewed as finding a median partition with respect to the given partitions in the ensemble, but determining this median partition has been shown to be NP-complete [2].

There are many types of consensus functions that solve this problem heuristically.Most of them require the number of clusters to be specified a priori, but in practice the number of clusters is usually unknown. In this paper, we propose

a new consensus function for cluster ensembles based on swarm intelligence [3] that addresses this problem. In particular, given a set of partitions, we apply ant clustering to the co-association matrix computed from the ensemble to produce the final partition, and automatically determine the number of clusters.

Ant clustering algorithms are inspired by how ants organize their food in their nests. Ant clustering typically involves two key operations: picking up an object from a cluster and dropping it off into another cluster. At each step, some ants perform pick-up and drop-off based on some notions of similarity between an objects and the clusters. In our method, the similarity measure is defined based on the co-association matrix. The clustering process is completely decentralized and self-organized, allowing the clustering structure to emerge automatically from the data. As a result, we can accurately determine the number of clusters in the data. The experimental results show that the proposed consensus function is very effective in predicting the number of clusters and also achieves reliable clustering performance. In addition, by introducing some simple heuristics, we can detect the marginal and outlier samples in the data to improve our final clustering.

The rest of the paper is organized as follows. Section 2 describes the basic concepts for cluster ensembles. The proposed consensus function and the heuristics for outlier and marginal sample detection are presented in Section 3. In Section 4, experimental results of the proposed method are compared with other methods. The paper is concluded in Section 5.

2 Cluster Ensemble Basics

Cluster ensembles have become increasing popular in the clustering community in recent years. A cluster ensemble framework usually consists of two stages. First, an initial set of partitions is generated to form the ensemble. It is typically required that the partitions are diverse, i.e., they need to be different from one another. Once a diverse set of partitions are generated, a consensus function is used to combine them and produce a final partition. Existing consensus functions can be grouped into the following categories:*Graph based methods* [1, 9], *Voting approaches* [4, 5],*Feature-based approaches* [6],*Co-association based methods* [7].

A common problem of the existing consensus functions is that, to apply techniques such as above, it is required that the number of clusters in the data be pre-specified. This can be problematic in practice because such information is rarely available. The method proposed in this paper addresses this problem by employing ant clustering to the co-association matrix, which requires no prior knowledge of the number of clusters. Let D be a data set of N data objects in a d-dimensional space. Let $P = \{P_1, \cdots, P_B\}$ be a set of partitions, where P_i is a partition of a bootstrap sample of D (or D itself) represented by X_i. In other words, each component partition in P is a set of clusters, i.e. $P_i = \left\{C_1^i, C_2^i, ..., C_{k(i)}^i\right\}$, where $C_1^i \cup C_2^i \cdots \cup C_{k(i)}^i = X_i \quad (X_i \subset D)$ and $k(i)$ is the number of clusters in the i_{th} partition.

The co-association matrix is computed as follows:

$$co - association(x, y) = \frac{1}{B} \sum_{i=1}^{B} \phi\left(P_i(x), P_i(y)\right)$$
$$\phi(a, b) = \begin{cases} 1 & \text{if} \quad a = b \\ 0 & \text{if} \quad a \neq b \end{cases} \tag{1}$$

Note that, in this paper we use the k-means algorithm with different initialization as our base clustering algorithm to generate the initial partitions and co-association matrix.

3 The Proposed Method

In ant based clustering approaches, each ant is an agent with simple probabilistic behaviors of picking up and dropping down objects (analogous to organizing their food in their nests) [10]. These probabilities depend on the similarity between the objects and other objects. Note that ant clustering has been typically applied using Euclidean distance to compute the similarity between objects. In our work, instead, we use the co-association matrix as the similarity measure among objects to perform ant clustering.

3.1 Ant Clustering Based on Co-association Matrix

Ant clustering begins by considering each object as a singleton cluster. The algorithm then proceeds by letting a fixed number of ants randomly move the objects around according to some basic probabilistic rules. There are two types of operations that the ants perform: picking up and dropping off objects. During clustering, each ant randomly selects a cluster from all clusters and picks up a sample from it and then drops it into an appropriate cluster. Below we explain the two types of operations.

Picking up: To perform a pick up operation, an ant first selects a cluster randomly from all existing clusters. If the selected cluster is empty, the ant will continue selecting until a non-empty cluster is found. Examining the selected cluster, there will be three possibilities: 1) the cluster contains only one object, 2) the cluster contains two objects and 3) the cluster contains more than two objects. In the first case, the ant picks up that object. In the second case, the ant selects one of the two objects randomly. In the final case, the ant identifies the object that is most dissimilar to the other objects in the cluster, which is defined as follow: For each sample x_i in cluster j we calculate:

$$S(x_i) = \frac{1}{\mid C_j \mid} \sum_{t \in C_j} co - association(x_i, t) \tag{2}$$

where $x_{Dissimilar}$ minimize S, i.e:

$$x_{Dissimilar} = argmin\ S(x_i) \qquad x_i \in C_j \tag{3}$$

If $S(x_{Dissimilar})$ is less than P_{remove}, the ant pick up $x_{Dissimilar}$. We set P_{remove} to be 0.5 which means that the ant will pick up $x_{Dissimilar}$ if it was clustered together with the other objects in the cluster for fewer than half of the original partitions of the ensemble. If the ant fails to find a qualifying object to pick up from the selected cluster, it will continue to select another cluster. Conceptually, the above probabilistic pick-up rule ensures that the ant always picks up objects that do not fit in the cluster well according to the co-association matrix, leading to clusters with the highest inner similarity.

Dropping off: After an ant picks up an object, it will look for an appropriate cluster to drop the object off. To do so, it first randomly selects a cluster from all available clusters. Depending on the number of objects contained in that cluster, there will be again three possible cases: 1) the cluster is empty, 2) the cluster contains one or two objects, and 3) the cluster contains more than two objects. In the first case, the ant does not drop the object and continues looking for another cluster. In the second case, the ant will only drop off the object if its average similarity with the objects(s) in the cluster in question is greater than P_{Create}. Here we set the P_{Create} to be 0.5, which requires the object to have been clustered together with the objects of the current cluster in more than half of the initial partitions of the ensemble. In the final case where there are more than three objects, the ant will only drop off the object if the object is more similar to the cluster than its current most dissimilar object. More specifically, its average similarity with the objects contained in the cluster in question needs to be greater than the average similarity between the most dissimilar object $x_{Dissimilar}$ and the other objects in the cluster. When this is true, it indicates that the carried object is more similar to the cluster than the most dissimilar object of the cluster is, providing evidence that the object should be part of that cluster. In all cases, if the ant fails to drop off the object in the selected cluster, the ant continues to look for another cluster. If the ant could not drop it in any cluster, it returns the object to its original cluster.

Sweeping: Due to the stochastic nature of the algorithm, it is possible to have some objects that have never been selected by any ant by the end of the clustering procedure. Further, some clusters may contain only a few objects, and should really be merged with other clusters to form bigger clusters. To address this issue, we include in our ant clustering algorithm a procedure called *sweeping* that examines all objects once and assigns them to an appropriate clusters. In particular, for all clusters that are singletons or contain too few examples (less than 5% of the examples), we assigns each of their objects into its most similar cluster based on the co-association values. Sweeping can effectively eliminate single-object clusters as well as clusters that consist of only a few objects. Note

that in total, our method uses $6*n$ ants, where n equals the number of objects in the data. We also interleave the regular ant clustering stage with the sweeping stage to address the problems caused by singleton and small clusters.

Algorithm Parameters: Traditional applications of ant clustering require many initial parameters which need insights about the data [10], and the clustering performance is highly sensitive to these parameters. In contrast, our method avoids this problem by working with the co-association matrix, whose element values are from a fixed range and have natural interpretations, making it easier to set the parameters in a meaningful manner such as described above. In addition, our method has only two critical parameters P_{Remove} and P_{Create}.

Recall that P_{Remove} is the threshold for deciding if an ant can picks up an object from a cluster of two or more objects. This parameter is intended to recognize the fact that sometimes all objects in a cluster are very close to one another and even the most dissimilar object is not far from the other objects. In such cases, it is intuitive to avoid picking up any object from such a tight cluster. Here P_{Remove} is set to 0.5, meaning that the most dissimilar object will not be picked up if it has been clustered together with other objects in more than half of the initial clusterings.

Similarly, P_{Create} is the threshold for deciding whether the ant can drop off an object to a cluster or not. We set P_{Create} to 0.5 which conceptually requires the object to have been clustered together with the other objects in more than half of the initial clusterings. The value 0.5 is completely understandable since we expect to cluster the sample with the objects that have been clustered together in more than half of the initial partitions.

It should be noted again that determining the initial parameters for traditional ant clustering algorithms can be challenging because the parameter values are based on the Euclidean distances and different data sets may have drastically different value ranges. Our method avoids this problem by working with the co-association matrix, whose element values are from a fixed range and have natural interpretations, making it easier to set the parameters in a meaningful manner such as described above.

3.2 Outlier and marginal objects detection

We distinguish between outlier and marginal examples in data. In particular, we refer to the objects that are far from the center of their clusters and also other objects in their clusters as outlier examples. In contrast, the marginal objects are those objects that border two or more clusters and as a result change their cluster memberships frequently in the ensemble. The outlier and marginal objects can be detrimental to clustering because they may mislead the algorithm. For example, outlier examples may make the estimation of cluster center inaccurate, and marginal objects may cause two or more clusters to be merged together incorrectly. Therefore, appropriately detecting the marginal and outlier objects can be useful in obtaining a better final partition. There have been numerous studies

on detecting outlier and marginal objects in data mining applications [8, 11]. In this paper we introduce some simple heuristics based on the ant clustering algorithm to effectively identify both outlier and marginal examples appropriately to help improve the final clustering.

Recall that the marginal objects are located at the intersecting areas of clusters and they frequently change their cluster associations in a cluster ensemble. Therefore, it is expected that their average similarity values with other objects in their own cluster to be low in comparison to other objects in their cluster. As a result, the marginal objects will likely be picked up frequently by ants during ant clustering. Furthermore, the ants will also likely fail to find an alternative cluster to place these marginal objects because they tend to be dissimilar to other clusters as well. We design our heuristic based on the above intuition. In particular, for each object we computes a marginality index that records how many times the object was picked up and then returned to the original cluster during ant clustering. The larger the marginality index, the more likely we consider the object to be marginal. It is clear that this procedure can be done along with the proposed algorithm without introducing any additional computational cost.

The outlier objects are those that are far from their cluster centers. They may lead to biased estimation of the cluster centers, resulting in some errors in clustering. The behavior of outlier objects is different from marginal examples in the sense that they usually are consistently assigned to specific clusters and rarely change their clusters. This means that if we investigate the co-association matrix, we expect an outlier object to have high similarities with most of the objects in its clusters, which is different from what we expect for the marginal examples. To detect the outlier objects, we add some virtual ants which deploy a different pick up strategy. In particular, these virtual ants pick up a sample from a cluster which is far from the center of the cluster based on the Euclidean distance instead of the co-association matrix values. These ants, however, will use the same drop off procedure as described before. If an object that is picked up by an virtual ant could not be dropped in any cluster and has to be returned to its original cluster it is likely an outlier. To capture this, we compute an outlier index for each object that records the number of times that it is picked up by a virtual ant and returned to its original cluster. The higher the value, the more likely we consider it an outlier.

In our experiments, the marginality indices are computed during the regular ant clustering procedure. The outlier indices are computed by applying the same ant clustering algorithm with the revised pick-up strategy with $6*n$ ants, where n is the number of objects in the data set. To designate an object as marginal or outlier, we require its corresponding index to be greater than a given threshold. Note that we observe a strong modal distribution of the index values, with most of the objects having index value zero and some with significantly higher value. This makes it easy to determine a good threshold. In our experiments, we set the threshold to be six for detecting both marginal and outlier examples. It should also be noted that the results of our algorithm are not sensitive to this threshold.

4 Experimental Results

We performed experiments on eight data sets. Four of them are from the UCI machine learning data sets [12] including the Iris, Wine, Soybean and Thyroid data sets and a real data set O8X from image processing data sets [13]. We also artificially generated three synthetic data sets. All data sets are preprocessed to normalize the feature values to a fixed range. This is done by dividing the value of each feature by the maximum value of that feature in all samples. Note that all data sets are labeled, but the labels are omitted during clustering and only used for evaluation purposes. In addition, to build the co-association matrix we use 100 independent runs of the k-means algorithm as initial clustering algorithm with different initializations.

Note that when evaluating the first two aspects, we do not apply the outlier/marginal sample detection heuristic and cluster all of the data. So the results presented in the following two subsections (4.1 and 4.2) are all based on the original data with no outlier/marginal examples removed. Furthermore, all reported results are averaged across 100 independent runs.

4.1 Identifying the Number of Clusters

In Table 1, we present the results of our method on predicting the number of clusters. In particular, the first column lists the name of each data set and the second column provides the real number of clusters (classes) of each data set. The third column shows the average number of clusters obtained by our algorithm and the fourth column shows the most frequent number of clusters from 100 independent runs. From these results, it can be seen that our method can usually obtain the number of clusters correctly.

We also compare our method with the ISODATA algorithm [13] which has been designed to obtain the number of clusters automatically. The ISODATA is a single clustering algorithm which can also detect the outlier samples based on its initial parameter values. The results of the ISODATA algorithm are presented in Table 2. The comparison indicates that our method is more effective at correctly determining the number of clusters than the ISODATA algorithm on the data sets used in our experiments. This is possibly due to the fact that ISODATA requires some initial parameters to be set appropriately to have a good clustering result while the proposed method has only two parameters and requires no fine tuning.

4.2 Clustering Accuracy

In Table 1, we show the misclassification error rates of the introduced method in column five. Also provided in the table are the results of two competitive consensus functions, namely the CSPA method (column six) and the average link agglomerative (ALA) method (column seven). Note that CSPA and ALA require the number of clusters to be set *a priori*. In our experiments, for these

Name	# of classes (Real)	# of classes (Average)	# of classes (most Frequent)	Proposed method error %	CSPA error %	ALA error %
O8X	3	2.92	3	9.33	11.2	8.7
Thyroid	3	3.1	3	14.50	15.86	48.65
Iris	3	2.9	3	5.80	4	5.5
Wine	3	2.85	3	7.10	9.88	10.11
Soybean	4	3.8	4	6.30	13.51	4.51
Artificial 1	2	2.2	2	4.50	5.5	4.2
Artificial 2	3	3.5	4	4.20	6.5	5.1
Artificial 3	3	2.9	3	6.30	15.91	8.6

Table 1. The proposed method results.

two methods, we set k to be the true number of classes in the data. The comparison indicates that the proposed method produced highly competitive clustering results. Note that we also present the error rates of the ISODATA algorithm in

Name	# of classes (Real)	# of classes (Average)	Miss classification error %
O8X	3	5.51	9.5
Thyroid	3	7.24	8.23
Iris	3	6.52	6.93
Wine	3	9.36	5.42
Soybean	4	8.34	2.72
Artificial 1	2	4.2	5.3
Artificial 2	3	5.5	7.3
Artificial 3	3	6.3	8.4

Table 2. The ISODATA algorithm results.

Table 2 (column four). While it appears that the ISODATA method outperforms all the other methods including the proposed method in a few data sets, this is due to the bias of the performance measure we used. In particular, the clustering results produced by ISODATA generally contain significantly larger number of clusters (as suggested by column 3 of Table 2) than the proposed method. Note that the classification error rate will always decrease as we increase the number of clusters in the clustering result. (Consider the extreme case where each point is in its own cluster, the error rate will be decreased to 0.) The readers are advised to interpret the results of ISODATA with caution.

4.3 Outlier and Marginal Detection

In this section, we evaluate the effectiveness of the introduced heuristics for identifying outlier and marginal examples from data. In Table 3 we show the scatter plot of the three artificial data sets and the Iris data, where the identified

outlier and marginal examples are highlighted as dark points. For Iris, we set feature 2 of the original data as the X axis and feature 3 as the Y axis to produce a 2-dimensional scatter plot.

The first row of Table 3 (a-d) highlights the samples that are identified as outlier in each data set. We can see that samples that are far from the center of the clusters have been selected as outliers. In the Iris data set this is less evident possibly because we only used two of the four features to produce the scatter plot.

We can see the identified marginal samples highlighted in the second row of Table 3 (e-h). We can see that the examples that are identified by our algorithm as marginal are primarily located in the area that borders multiple clusters. In the Iris data set, which consists of three clusters, the marginal sample are mainly located in the area between two of the clusters since the third cluster is linearly separable from the other two clusters. In the Artificial 1 and 2 dataset the samples which are located between clusters have been selected as marginal samples and in Artificial 3 data set the samples which have been located in the conjunction of the three clusters have been selected as marginal samples. It can be seen that the selected marginal samples are different from the outlier samples.

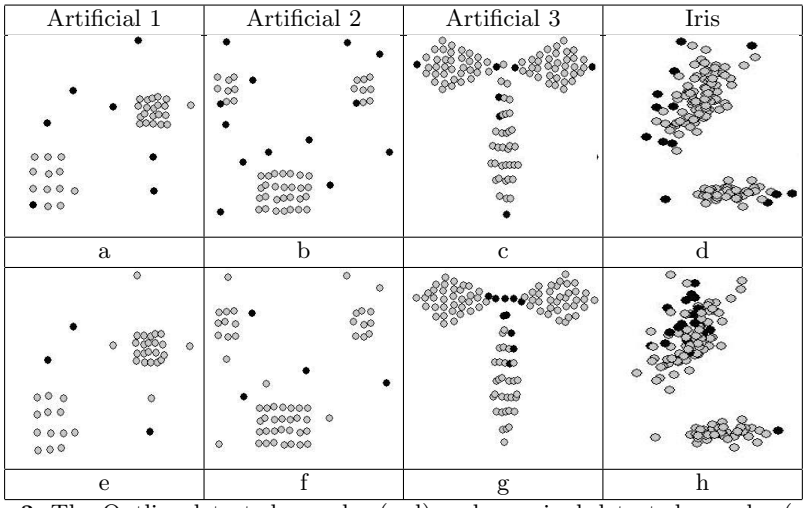

Table 3. The Outlier detected samples (a-d) and marginal detected samples (e-h) in related data sets.

5 Conclusion

In this paper, we presented an ant colony based algorithm which forms clusters from the results of other clustering methods. The input to our method is an

ensemble of various clusterings. The proposed method applies ant clustering to the co-association matrix of the ensemble to produce the final partition. Our method has the unique ability to automatically determine the number of clusters. Furthermore, it produces reliable clustering performance. Compared to other applications of ant clustering, our method uses fewer parameters than other ant based clustering methods . The use of the co-association matrix instead of Euclidean distance allows us to avoid fine tuning of the parameters. Finally, a by-product of our ant clustering algorithm is that we can introduce very simple heuristics that can effectively identify both marginal and outlier examples, which in turn further improves the performance of the final partition. Our experimental results provide clear evidence that the proposed method can: 1) reliably predict the number of clusters in the data, 2) produce highly competitive clustering results in comparison to other ensemble methods, and 3) effectively identify marginal and outlier examples.

References

1. Strehl, A., Ghosh, J.: Cluster ensembles-a knowledge reuse framework for combining partitioning. In: IJCAI, Edmonton, Alberta, Canada, pp. 93–98 (2002).
2. Barthelemy, J.P., Leclerc, B.: The median procedure for partition. Partitioning Data Sets, AMS DIMACS Series in Discrete Mathematics,3–34 (1995).
3. Kennedy,J., S. Russell., Yuhui,S.: Swarm Intelligence. Morgan Kaufmann. San Francisco, California.(2001).
4. Fern, X.Z, Brodley: Random projection for high dimensional data clustering: a cluster ensemble approach. In: ICML, Washington, DC, pp.186-193 (2003).
5. Weingessel, A., Dimitriadou, E., Hornik, K.: An ensemble method for clustering. Working paper, `http://www.ci.tuwien.ac.at/Conferences/DSC-2003`, (2003).
6. Topchy, A., Jain, A.K., Punch, W.: A mixture model for clustering ensembles. In: Proceedings of SIAM Conference on Data Mining, pp.379–390 (2004).
7. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd Edition. John Wiley and Sons Inc. New York NY (2001).
8. Topchy, A., Minaei-Bidgoli, B., Jain, A.K., Punch, W.: Adaptive Clustering ensembles. In: International Conference on Pattern Recognition, Cambridge, UK, pp. 272–275 (2004).
9. Fern, X.Z, Brodley, C.E.: Solving cluster ensemble problems by bipartite graph partitioning. In: 21th International Conference on Machine Learning, pp. 281–288 (2004).
10. Monmarche, N., Silmane, M., Venturini, G.: AntClass:discovery of clusters in numeric data by an hybridization of an ant colony with k-means algorithm. Internal Report no. 213, Laboratoire 'Informatique de l'Universite (1999).
11. Jain, A.K., Murty, M.N., and Flynn, P.J.: Data clustering: A review. ACM Comp. Surveys, 31(3), 264–323 (1999).
12. Blake, C., Merz, C. UCI repository of machine learning databases. `http://archive.ics.uci.edu/ml/`, (1998).
13. Gose, E., Johnsbaugh, R., Jost, S.: Pattern Recognition and Image Analysis. Prentice. Hall (1996).