International Journal of Machine Learning and Cybernetics
https://doi.org/10.1007/s13042-017-0756-7

ORIGINAL ARTICLE

Clustering ensemble method

Tahani Alqurashi[1] · Wenjia Wang[1]

Received: 28 September 2015 / Accepted: 20 October 2017

Abstract
A clustering ensemble aims to combine multiple clustering models to produce a better result than that of the individual clustering algorithms in terms of consistency and quality. In this paper, we propose a clustering ensemble algorithm with a novel consensus function named Adaptive Clustering Ensemble. It employs two similarity measures, cluster similarity and a newly defined membership similarity, and works adaptively through three stages. The first stage is to transform the initial clusters into a binary representation, and the second is to aggregate the initial clusters that are most similar based on the cluster similarity measure between clusters. This iterates itself adaptively until the intended candidate clusters are produced. The third stage is to further refine the clusters by dealing with uncertain objects to produce an improved final clustering result with the desired number of clusters. Our proposed method is tested on various real-world benchmark datasets and its performance is compared with other state-of-the-art clustering ensemble methods, including the Co-association method and the Meta-Clustering Algorithm. The experimental results indicate that on average our method is more accurate and more efficient.

Keywords Clustering ensemble · K-means · Similarity measurement · Machine learning · Data mining

1 Introduction

In the context of machine learning, an ensemble is generally defined as "a machine learning system that is constructed with a set of individual models working in parallel, whose outputs are combined with a decision fusion strategy to produce a single answer for a given problem" [44]. The ensemble method was firstly introduced and well-studied in supervised learning fields. Due to its successful applications in classification tasks, in the past decade or so, researchers have attempted to apply the same paradigm to unsupervised learning fields, particularly clustering problems, for two obvious reasons. Firstly, in unsupervised learning, there is normally no prior knowledge about the underlying structure or about any particular properties that we want to find or about what we consider good solutions for the data [23, 38]. Different clustering algorithms may produce different clustering results for the same data by imposing a particular structure onto the data. Secondly, there is no single clustering algorithm that can perform consistently well for different problems and there are no clear guidelines to follow for choosing individual clustering algorithms for a given problem.

Conceptually speaking, a clustering ensemble, also referred to as a consensus ensemble or clustering aggregation, can be simply defined in the same manner as for classification, that is, the process of combining multiple clustering models (partitions) into a single consolidated partition [36]. In principle, an effective clustering ensemble should be able to produce more consistent, reliable and accurate clustering results compared with the individual clustering algorithms.

However, the transmission from supervised learning to unsupervised learning is not as straightforward as this conceptual definition because there are some unique and challenging issues when building an ensemble for clustering. Out of these issues, the key and most difficult one is how to combine the clusters that are generated by the individual clustering models (members) in an ensemble, as this cannot be done through simple voting or averaging as in classification–it requires more complicated aggregating strategies and mechanisms. There is no effective and scalable consensus function in practical application yet, although many have been proposed to date.

✉ Tahani Alqurashi
toand2@hotmail.com

Wenjia Wang
wenja.wang@uea.ac.uk

[1] School of Computing Science, University of East Anglia, Norwich Research Park, Norfolk, UK

Published online: 16 January 2018

In this paper, we propose a three-staged adaptive consensus function based on two similarity measures and use it to build a clustering ensemble framework, named the Adaptive Clustering Ensemble (ACE). The first stage is to transfer the members into binary representation, the second stage is to measure the similarity between the initial clusters and adaptively merge the most similar ones to produces k consensus clusters. The third stage is to identify the candidate clusters that contain only certain objects and to calculate their quality. The final clustering result is produced by an iterative process assigning the uncertain objects to a cluster in a way that has a minimum effect on its quality.

This is in fact an improved version of our earlier work [2], where we developed a Dual-Similarity Clustering Ensemble method (DSCE). The DSCE algorithm has been improved in three aspects. Firstly, the stability of the DSCE has been improved by producing the final clustering result with the pre-defined k, even when the members have a different number of clusters. Secondly, the effect of its two parameters (α_1 and α_2) on the quality of the final result has been reduced by applying an adaptive strategy for the value for these thresholds. Finally, the object neighbourhood similarity for the uncertain objects has been taken into account, in order not to lose any information when we eliminate an inappropriate cluster.

The rest of the paper is organised as follows. Section 2 introduces the clustering ensemble problem and the general clustering ensemble framework. Section 3 summarises the related work, while Sect. 4 details the proposed clustering ensemble method with its different stages. Section 5 discusses the experimental studies and Sect. 6 shows the results on real datasets. Section 7 presents the parameter analysis and the time complexity analysis of the proposed method. Finally, conclusions are given in Sect. 8.

2 Clustering ensemble methods

2.1 Clustering ensemble representation

For a dataset of n objects: $X = \{x_1, x_2, \dots, x_n\}$, let $P_q = \{c_1^q, c_2^q, \dots, c_{k_q}^q\}$ be a clustering result of k_q clusters produced by a clustering algorithm as q^{th} partition, so that $c_i^q \cap c_j^q = \emptyset$ and $\cup_{j=1}^{k_q} c_j^q = X$. A clustering ensemble $\boldsymbol{\Phi}$ can then be built with m partitions $\Gamma = \{P_1, P_2, P_3, \dots, P_m\}$ and a consensus function F, and denoted by $\boldsymbol{\Phi}(F, \Gamma) = F(P_1, P_2, P_3, \dots, P_m) = F(\Gamma)$. It should be noted that the members may not necessarily have the same number of clusters in their partitions, that is, k_q may not be equal to a pre-set value k.

The task of a clustering ensemble is to find a partition P^* of dataset X by combining the ensemble members $\{P_1, P_2, P_3, \dots, P_m\}$ with F without accessing the original features, so that P^* is probably better in terms of consistency and quality than the individual members in the ensemble.

2.2 A generic clustering ensemble framework

A common clustering ensemble framework is represented in Figure 1, which consists of three components: ensemble member generation, consensus function and evaluation. As can be seen, the input of the clustering ensemble framework is a given dataset to be clustered, and the output is the final clustering result of this dataset.

2.2.1 Ensemble member generation

This is the first phase in the clustering ensemble framework, and the main aim here is to generate m clustering models as the members for building the ensemble. In principle, any clustering algorithm could be used here as long as it is suitable for the dataset. In addition, the generated members should be different to each other as much as possible, because a high level of diversity among the members means that they have captured different information about the data and can potentially help to improve the performance of the ensemble. Thus, it is important to apply one or several appropriate generation techniques to achieve reasonable quality as well as diversity.

Some researchers have selected techniques based on the type of applications. For example, for high dimensional data, Strehl and Ghosh [36] used random feature subspaces and members are generated for each subspace. They

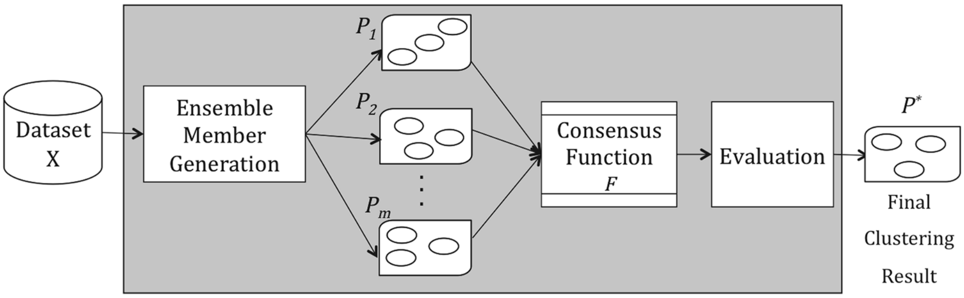


Fig. 1 A generic clustering ensemble framework

also generated members by selecting different subsets of objects for each member, and they called this technique object distribution. Fern and Brodley [8] generated members based on random projections of objects into different subspaces.

The resampling method was also used by [3, 30, 31]. In particular Minaei-Bidgoli et al. [30] used bootstrap techniques with a random restart of *k-means*, while Monti et al. [31] used bootstrap techniques with different clustering algorithms, including *k-means*, model-based Bayesian clustering and self-organising maps. Moreover, Ayad and Kamel [3] used the bootstrap resampling in conjunction with *k-means* to generate the ensemble members.

Arguably, the most commonly used clustering algorithm for generating members is *k-means* because of its simplicity and low computational complexity [4, 10–12, 20, 39]. For instance, Fred and Jain [11] used the *k-means* clustering algorithm with random initialisations of cluster centres and a randomly chosen k (number of clusters) from a pre-specified interval for each member, and they used a large k value in order to obtain a complex structure within the ensemble members. They also ran *k-means* with a fixed k to compare the two generation techniques, and they found that members with a random k are more robust than other members. Dimi-triadou et al. [6] and Sevillano et al. [34] applied fuzzy clustering algorithms, and in particular c-means in order to generate soft clustering members, while Hore et al. [14] applied fuzzy *k-means*.

Strehl and Ghosh [36] used a graph-clustering algorithm with different distance functions for each member. Topchy et al. [40] used a weak clustering algorithm, which produces a clustering result that is slightly better than a random result in terms of accuracy due to the fact that it uses a random projections on one dimension and splitting the data into a random number of hyperplanes. Iam-on et al. [19] examined different techniques, which included a multiple run of *k-means* with a fixed k for each member, and randomly chosen k from an interval, where the maximum k was equal to $\sqrt{n}$. Furthermore, Iam-On et al. [21] applied different generation techniques to categorical data, and they ran a *k-mode* algorithm with full space and random subspace with a fixed k and random k. They found that these two techniques allowed their ensemble method to achieve high performance, compared to the *k-mode* clustering algorithm, as well as several other ensemble method such as methods proposed by Strehl and Ghosh [36].

Another strategy is to use a different clustering algorithm for each member [12, 45] with a hope that different algorithms may generate more diverse members. Yi et al. [45] used some well-known clustering algorithms, such as hierarchical clustering and *k-means*. Gionis et al. [12] used the single, average, ward and complete linkage methods and *k-means* to generate ensemble members.

In summary, as can be seen, there is no single clustering algorithm that is universally used and there are no generally agreed criteria for selecting the most suitable ones. In this case, it is better to apply the principle of Occam's razor [5] and choose the one with the greatest simplicity and efficiency, if there is no prior specific knowledge on a given problem. This is why we chose *k-means* over others in our experiments in this study.

2.2.2 Consensus function

A consensus function combines the outputs of the members $\{P_1, P_2, P_3, \dots, P_m\}$ to obtain the final clustering result P^*, and can directly determine the quality of the final solution. Therefore, it is considered the most important component in an ensemble. A number of existing consensus functions have been reviewed by Vega-Pons and Ruiz-Shulcloper [41] and they are classified into two main approaches: object co-occurrence and median partition.

The object co-occurrence approach: It firstly computes the co-occurrence of objects in the members and then determines their cluster labels to produce a consensus result. Simply, it counts the occurrence of an object in one cluster, or the occurrence of a pair of objects in the same cluster, and generates the final clustering result by a voting process among the objects. Such methods are the Relabelling and Voting method [4, 7, 47], the Co-association method [11] and the Graph-based method [9, 36].

The median partition approach: This treats the consensus function as an optimisation problem of finding the median partition with respect to the cluster ensemble. The median partition is defined as "the partition that maximises the similarity with all partitions in the clustering ensemble" [41]. Examples of this approach include the Non-Negative Matrix Factorisation based method [26], the Genetic-based method [28, 46] and the Kernel-based method [42].

Vega-Pons and Ruiz-Shulcloper [41] pointed out that consensus functions were primarily studied on a theoretical basis, and as a result many consensus functions based on the median partition approach were proposed in the literature, whereas only a few studies focused on the object co-occurrence approach. Therefore we chose to develop a consensus function based on the object co-occurrence approach, and for this reason, we will only review the work related to this approach in Sect. 3.

2.2.3 Evaluation

In this phase, the aim is to evaluate the quality of the final clustering result. Evaluating the quality of clustering results is a non-trivial task as there is no universally agreed standard on what constitutes good quality clusters in the first place. There are a number of aspects that need to be considered when evaluating the clustering result, but in practice, the most common ones are probably accuracy and consistency. For measuring accuracy, there are many external validation indexes or measures that can be used to evaluate the accuracy, but the most common ones used in clustering ensemble research are the Adjust Rand Index (ARI) [18] and the Normalised Mutual Information (NMI) [36]. For consistency, it is usually represented by the average of a performance measure and its variance (e.g. standard deviation) [1] from repeated runs with different experimental set-up conditions.

3 Related work

The most popular method in the object co-occurrence approach compares the cluster association of each object and produces a pairwise object similarity matrix, called an adjusted similarity matrix [36], consensus [31], agreement [37] and Co-association matrix [11], then the final partition is obtained by applying the hierarchical clustering algorithm. But, perhaps, any similarity-based clustering algorithm can be applied to this matrix.

The Co-association method (CO) avoids the label correspondence problem by mapping the ensemble members onto a new representation in which the similarity matrix is calculated between a pair of objects in terms of how many times a particular pair is clustered together in all ensemble members [11]. Basically, CO calculates the percentage of agreement between ensemble members in which a given pair of objects is placed in the same cluster as follows:

$$CO(x_i, x_j) = \frac{1}{M} \sum_{m=1}^{M} \delta(P_m(x_i), P_m(x_j))$$

Where x_i and x_j are objects, P_m is a partition, and δ is defined as:

$$\delta = \begin{cases} 1, & \text{if } x_i \text{ and } x_j \text{ are placed in the same cluster.} \\ 0, & \text{otherwise.} \end{cases}$$

In Fred and Jain [11], the final partition is obtained by applying Single and Average linkage hierarchical clustering algorithms to the Co-association matrix. The CO seems ideal for collecting all the information available in the clustering ensemble, but in fact it takes into consideration just the pairwise relationship between objects in the ensemble members. Strehl and Ghosh [36] proposed the Cluster-based Similarity Partitioning Algorithm (CSPA), where also the object pairwise similarity was taken into account by representing it as a fully connected graph, where nodes correspond to data objects and edge weights to their similarities. The final clustering results are obtained by applying the METIS algorithm [24] to the constructed graph.

An alternative method to the object pairwise similarity matrix is to consider the association between object and cluster, which is formulated as a binary membership matrix [9, 36]. Fern and Brodley [9] represent this object-cluster membership matrix as a bipartite graph, which is called a Hybrid Bipartite Graph Formulation (HBGF) algorithm. In this graph, there are two different types of nodes; one represents an object and the other represents a cluster, and an edge exists between a cluster and an object belonging to that cluster. Then a spectral clustering algorithm was applied to obtain the final partition. Strehl and Ghosh [36] proposed the hypergraph partitioning algorithm (HGPA), and the Meta-CLustering Algorithm (MCLA). The hypergraph is constructed in HGPA and MCLA, where each cluster is represented as a hyperedge. HGPA directly partitions the hypergraph by cutting a minimal possible number of hyperedges into k connected nodes of approximately the same size using the hypergraph partitioning package HMETIS [24]. MCLA firstly defines the similarity between pair clusters in terms of the shared objects between them, using the extended Jaccard index [35]. The graph is then constructed where nodes represent clusters and the edges represent the similarity between pairs of clusters. The final partition, 'meta-clustering', is obtained using METIS [24]. The complexity of CSPA, HGPA and MCLA is estimated as $O(kn^2m)$, $O(knm)$, and $O(k^2nm^2)$ respectively [36].

A further development by Iam-On et al. [20] aimed to redefine the Co-association matrix to also take into account the relationship between clusters estimated from a link network model, and to interpret this matrix as feature vectors or a Bipartite graph. It used an ordinary clustering algorithm onto the new similarity matrix in order to generate the final cluster result.

Another method named 'Division Clustering Ensemble' (DICLENS) was developed by Mimaroglu and Aksehirli [29], based on minimum Spanning Tress Similarity, where each vertex represents a cluster and the edge represents the inter-cluster similarity between clusters. They are, in fact, redefining the co-association matrix to be calculated between two objects placed in two different clusters, which represents the inter-cluster similarity. They then cut edges with the lowest similarity to produce disjoining meta-clusters, which represent the final clusters where each object is assigned to the most associated cluster. It should be noted that they ran the experiment just once using only manually generated members. It is widely known that the generated members have a direct and strong influence on the ensemble performance and, in many real-world applications, it is impossible or impractical to generate clusters manually, so then a clustering algorithm has to be employed to generate clustering members automatically.

Recently, Alqurashi and Wang [1] highlighted a problem relating to the uncertain agreement between members.

A new method named 'Object-Neighbourhood Clustering Ensemble' (ONCE) was proposed by taking into account the neighbourhood relationship between object pairs, as well as the relationship between the pair itself in the similarity matrix calculation. The uncertain agreement problem was also tackled by Huang et al. [17], who applied a sparse graph representation and the probability trajectory analysis to propose two clustering ensemble algorithms, named 'Probability Trajectory accumulation' (PTA) and 'Probability Trajectory based graph partitioning' (PTGP). In both algorithms, they first constructed K-elite neighbour sparse graph (K-ENG) and they calculated the probability trajectory similarity matrix. The random walk process was performed on the K-ENG graph to derive a dense similarity matrix based on the probability of random walkers. In PTA they applied hierarchal clustering algorithm to obtain the final clustering results, whereas in PTGP they applied the Tcut algorithm [27].

One of the main drawbacks of clustering ensemble methods, based on object pairwise similarity, is that they do not scale very well for a large dataset, as they work at the object level, and they do not capture the relationship between clusters. However, a clustering ensemble method based on the similarity between clusters, such as MCLA, is much faster than CO and CSPA. Another point is that most of the clustering ensemble approaches transform the initial clusters produced by the member into a new representation, and then produce the final clustering result by clustering this new representation with an ordinary clustering algorithm. When applying the same representation to a different clustering algorithm, their performance can vary considerably and it can be difficult to decide which clustering algorithm is the best one to use. Huang et al. [16] also highlighted this limitation and they proposed an algorithm named 'Ensemble Clustering using Factor Graph' (ECFG), which redefines the ensemble clustering problem into a binary linear programming problem and they solved this optimisation problem with a factor graph. ECFG first estimates the reliability of the clustering decisions of the members using an EM algorithm, and it has the ability to automatically generate the number of clusters in the final clustering result. In their experiment, Huang et al. [16] did not report the estimated number of clusters for each tested datasets and they used NMI as an evaluation measure, which is suitable for comparing two partitions that have equal number of clusters. We also think that they should validate their results using more than one evaluation measure.

Therefore, there is a gap in clustering ensemble methods with regard to considering the relationship between initial clusters, as well as between clusters and objects, and this is the motivation of this study.

4 The adaptive clustering ensemble (ACE)

As the consensus function plays a key role in a clustering ensemble, directly influencing its performance, our aim is to design a consensus function that is more effective and efficient. The main idea of the proposed consensus function is that, instead of calculating the similarity between a pair of objects (the object pairwise similarity) as in the CO method, we calculate the similarity between pairs of clusters generated by the members and we then derive the membership similarity between newly formed clusters and objects. The rationale is that we have already generated clusters in the first phase of the ensemble process, so it is obviously more efficient and possibly more effective to consider just the similarity between the initial clusters instead of object similarity. We can then extend the concept of shared-neighbour information from the object level to the cluster level. Therefore, two clusters are considered to be well-associated if their objects resemble one another to a certain degree. If two clusters have a high proportion of objects in common as determined by the ensemble members, they should be merged, whereas if two clusters have a smaller proportion of objects in common, they should be kept separated.

However, instead of following some of the single clustering algorithm procedures in building a consensus function, we are using the generated members as initial clusters of the dataset and the final clustering is generated in three stages, as shown in Fig. 2. The first stage is to transfer the members into a binary vector representation. The second is to generate the consensus clusters, where the similarity between initial clusters is measured and the predefined k clusters are produced. The third stage is to solve uncertain objects, where firstly a certain object is assigned to the cluster that has a higher membership value and then the uncertain objects are classified to the cluster in a way that has a minimum effect on the cluster quality. The developed algorithm is called the Adaptive Clustering Ensemble (ACE).

The following sections present the definitions of the similarity measures and terminologies and then explain in detail how the algorithm works in three stages.

4.1 Definitions of similarity measures

We define two similarity measures: similarity between clusters and similarity between objects and clusters. The latter is measured by the degree of membership by which an object belongs to a cluster, hence it is called membership similarity. Before defining these similarity measures, we introduce some notations used throughout this paper as follows:

1. S_c: The cluster similarity measure between two clusters.

2. S_x: The membership similarity measure.
3. θ_1: The membership matrix, where the columns of this matrix correspond to clusters and the rows correspond to objects.
4. δ: A binary membership value of an object to a particular cluster, $\delta \in \{0, 1\}$.
5. α_1: A threshold for merging clusters, its value is determined based on S_c.
6. α_2: A certainty threshold for placing an object into a cluster, its value is determined based on S_x: Number of clusters in θ_1.
7. C: The set of all the newly formed clusters after the merging process has concluded.
8. P_c: Cluster certainty, only calculated for a newly formed cluster.

Definition 1 *Cluster similarity*: Given an ensemble Φ that is built with m clustering partitions $\Gamma = \{P_1, P_2, P_3, \dots, P_m\}$ of dataset $X = \{x_1, x_2, \dots, x_n\}$, cluster similarity S_c is a measure of how much overlap there is between two clusters from different partitions.

We employ the 'set correlation' as a cluster similarity measurement, which measures the overlap between two clusters and takes their size into account. It has been developed in the Relevance-Set Correlation (RSC) [15] model, as this measure is an equivalent of the Pearson correlation in clustering analysis. After some simplification and derivation, it can be represented as follows:

$$
\begin{aligned}
S_c(c^q_{j_q}, c^\ell_{j_\ell}) &= \frac{|c^q_{j_q} \cap c^\ell_{j_\ell}| - \frac{|c^q_{j_q}||c^\ell_{j_\ell}|}{n}}{\sqrt{|c^q_{j_q}||c^\ell_{j_\ell}|\left(1 - \frac{|c^q_{j_q}|}{n}\right)\left(1 - \frac{|c^\ell_{j_\ell}|}{n}\right)}} \\
&= \frac{n.CM(c^q_{j_q}, c^\ell_{j_\ell}) - \sqrt{|c^q_{j_q}||c^\ell_{j_\ell}|}}{\sqrt{(n - |c^q_{j_q}|)(n - |c^\ell_{j_\ell}|)}},
\end{aligned} \tag{1}
$$

where q and ℓ are two members, $q \neq \ell$, and j_q, j_ℓ are the cluster index in q and ℓ respectively. CM is the Cosine similarity measurement [13]:

$$
CM(c^q_{j_q}, c^\ell_{j_\ell}) = \frac{|c^q_{j_q} \cap c^\ell_{j_\ell}|}{\sqrt{|c^q_{j_q}||c^\ell_{j_\ell}|}} \tag{2}
$$

S_c is symmetric, i.e. $S_c(c_i, c_j) = S_c(c_j, c_i)$ and its value is bounded in [− 1, 1]. A value of 1 indicates that the two clusters "are identical", and a value of − 1 indicates that the two clusters are "a complement of each other" [43].

Definition 2 *Membership similarity* In general, this is a measure of similarity S_x between an object x and a cluster c (when a soft clustering is allowed, i.e. an object x may be placed in more than one cluster), and hence it is defined as the membership similarity.

In this study, it is specifically used to measure the similarity between objects $x_i \in X$ in a new cluster, $\overleftarrow{c}_g$, that is formed by merging r (initial) clusters $\overleftarrow{c}_g = \{c_i + c_j + \cdots + c_r\} \in \Gamma$, so that $S_c(c_i, c_j, \dots, c_r)$ is higher than a pre-set threshold. It is defined as follows:

$$
S_x(x_i, \overleftarrow{c}_g) = \frac{1}{\max\{\theta_1(x_i, \overleftarrow{C})\}} \theta_1(x_i, \overleftarrow{c}_g), \tag{3}
$$

where, $\overleftarrow{C}$ is the set of all the newly formed clusters, $\overleftarrow{C} = \{\overleftarrow{c}_1, \dots, \overleftarrow{c}_g, ...\}$; $\theta_1(x_i, \overleftarrow{c}_g)$ is the membership of x_i belonging to cluster $\overleftarrow{c}_g$ and is defined as follows:

$$
\theta_1(x_i, \overleftarrow{c}_g) = \sum_{u=1}^{r} \delta(x_i, c_u) \tag{4}
$$

The value of membership similarity S_x is bounded between 0 and 1, and a higher value means a stronger membership or a higher degree of certainty that an object belongs to a cluster. Therefore, objects with different values of this measure can be classified as certain, uncertain or totally uncertain for a given threshold value α_2, as defined below.

Definition 3 *Certain object:* An object, x_i, is defined as a certain object if its maximum membership similarity S_x is greater than a pre-set value α_2, i.e.

$$
\max(S_x(x_i, \overleftarrow{C})) > \alpha_2. \tag{5}
$$

Definition 4 *Uncertain object:* An object is defined to be an uncertain object if its maximum membership similarity S_x is less than or equal to α_2, i.e.

$$
\max(S_x(x_i, \overleftarrow{C})) <= \alpha_2. \tag{6}
$$

Definition 5 *Totally certain object:* An object is defined as a totally certain object if its maximum membership similarity S_x is 1.

Definition 6 *Totally uncertain object:* An object is defined as a totally uncertain object if its maximum membership similarity S_x is 0.

Definition 7 *Cluster certainty:* The cluster certainty, ρ_{c_g}, is defined as the mean of the membership similarity of objects in a cluster $\overleftarrow{c}_g$, i.e.

$$\rho_{c_g} = \frac{1}{|\overleftarrow{c}_g|} \sum_{i=1}^{|\overleftarrow{c}_g|} S_x(x_i, \overleftarrow{c}_g). \tag{7}$$

The cluster certainty is calculated for each newly formed cluster $\in \overleftarrow{C}$.

Choosing an initial value for α_2 is not so critical as our ensemble algorithm adapts its value through its consensus function during the iterations of the algorithm. It is therefore reasonable to set the initial value for α_2 to be 0.7, and to then adapt it if necessary, according to the values of the updated membership similarity matrix S_x as described in the algorithm. The detailed investigation and analysis of its influence will be given in Sect. 7.

4.2 The ACE algorithm

The diagram of the ACE algorithm is given in Fig. 2 and as can be seen it works in three main stages: Transformation, Generating Consensus Clusters and Resolving Uncertainty. We will give a simple example to illustrate how this algorithm works throughout these stages.

4.2.1 Stage 1: transformation

Having generated m members, which represent unmatched clusters of objects, this stage transforms them into a new representation. In order to avoid solving the relabelling problem between clusters, we transform each cluster c to a column binary characteristic vector where a value of 1 indicates that the corresponding object belongs to that cluster, and 0 indicates that the object does not belong to that cluster.

In general, for cluster c_j in clustering member q, its corresponding vector is represented as $c_j^q = [\delta(x_1), \dots, \delta(x_n)]^T$, where $\delta(x_i)$ is the binary membership and takes the following value:

$$\delta(x_i, c_j) = \begin{cases} 1, \text{ if } x_i \in c_j, \forall\, i = 1, \dots, n. \\ 0, \text{ if } x_i \notin c_j \end{cases} \tag{8}$$

Where i is the index of data objects; $j(= 1, \dots, k_q)$, the index of clusters in each of m members; $q(= 1, \dots, m)$ is the index of members in an ensemble. There will be k_m vectors to form an $n \times k_m$ cluster matrix $\theta_1 = [c_1^1, c_2^1, \dots, \dots, c_{k_m}^q]$. Where $k_m = \sum_{q=1}^{m} k_q$, which is the total number of clusters in all members.

4.2.2 Stage 2: generating new consensus clusters

In this stage, the aim is to find the most similar initial clusters and to merge them to produce k clusters that are as dissimilar from each other as possible. To achieve this, the following two steps are required:

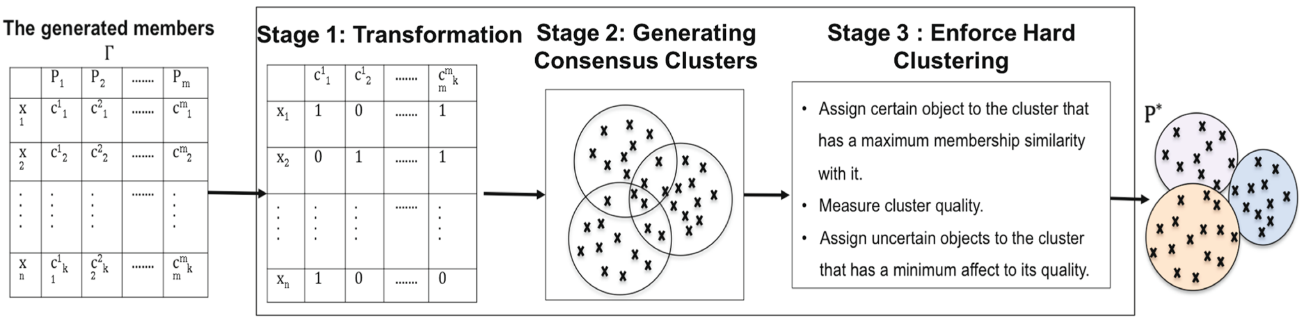
1. *Measuring similarity between initial clusters and merging the most similar ones*

   (a) Starting with k_m initial clusters, we measure the cluster similarity S_c, defined in Eq. 1, between the initial clusters that are placed in different members in $\boldsymbol{\Phi}$.

   (b) The merging process is performed based on the following criterion:

$$\text{if} \quad S_c(c_{j_q}^q, c_{j_\ell}^\ell) >= \alpha_1 \quad \Rightarrow \quad c_{j_q}^q \text{ and } c_{j_\ell}^\ell \text{ are similar, hence merged.} \tag{9}$$

$$\text{if} \quad S_c(c_{j_q}^q, c_{j_\ell}^\ell) < \alpha_1 \quad \Rightarrow \quad c_{j_q}^q \text{ and } c_{j_\ell}^\ell \text{ are dissimilar, not merged.} \tag{10}$$


The generated members
Γ
Stage 1: Transformation
Stage 2: Generating Consensus Clusters
Stage 3 : Enforce Hard Clustering
P*

	P_1	P_2	....	P_m
x_1	c^1_1	c^2_1	....	c^m_1
x_2	c^1_2	c^2_2	....	c^m_2
⋮	⋮	⋮	....	⋮
x_n	$c^1_{k_1}$	$c^2_{k_2}$	....	$c^m_{k_m}$

	c^1_1	c^1_2	....	$c^m_{k_m}$
x_1	1	0	....	1
x_2	0	1	....	1
⋮	⋮	⋮	....	⋮
x_n	1	0	....	0

- Assign certain object to the cluster that has a maximum membership similarity with it.
- Measure cluster quality.
- Assign uncertain objects to the cluster that has a minimum affect to its quality.

Fig. 2 The diagram of the ACE algorithm

Parameter α_1 is the threshold for merging and is determined adaptively based on the similarity values in the cluster similarity matrix S_c.

Its influence and sensitivity on the quality of the final clustering result are studied and the details are given later in Sect. 7. Our empirical study indicates that it can usually start with a relatively high value, e.g. 0.8, and then adapt its value in accordance with the similarity values in the current similarity matrix.

From S_c, any clusters that satisfy a criterion given in Eq. 9 will be merged by replacing them with a new cluster $\overline{c}_j$. This continues until there remain no pairs of clusters that are similar enough.

Then the membership similarity S_x between objects and the newly formed clusters are calculated using Eq. (4).

To illustrate these steps, we measured the similarity between the initial cluster vectors in our illustrative example Fig. 3, and gained the similarity matrix S_c. We set α_1 equal to 0.8. Looking at S_c, we found that c_1^1 and c_2^3 were identical and had a similarity greater than α_1 with c_2^2, so we merged them. In addition, c_2^1 had a similarity greater than α_1 with c_3^2 and c_1^3, so we merged them too. We also merged c_3^1 and c_1^2. As a result, we gained four clusters, $\overline{c}_1$, $\overline{c}_2$, $\overline{c}_3$ and $\overline{c}_4$ in θ_1. Thus, θ_1 become the input for the next step in this stage.

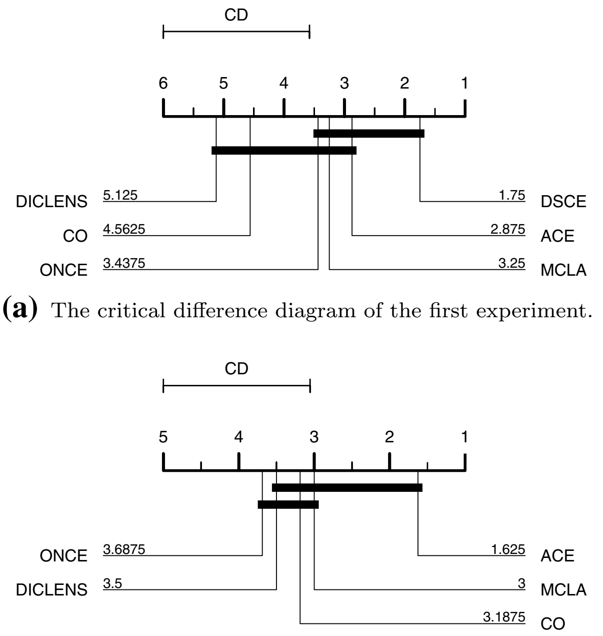
2. *Producing k consensus clusters*

   After the most similar initial clusters are merged, we have θ_1 to represent newly formed clusters and perhaps some remaining non-merged initial clusters. The next step is to check if the number of the clusters in θ_1 is exactly equal to k clusters, which will be taken as the final candidate clusters. For convenience, let λ be the number of clusters in θ_1. There are three possible scenarios: (1) $\lambda = k$, (2) $\lambda > k$, and (3) $\lambda < k$, when checking the number of clusters in θ_1.

   (a) When $\lambda = k$, i.e. the number of clusters in θ_1 is equal to the pre-defined k, we then take the clusters in θ_1 as the candidate clusters and adapt α_2 to a value based on S_x so that it can represent a specific percentage of the membership certainty. Then we move onto Stage 3.

   (b) When $\lambda > k$, i.e. the number of clusters in θ_1 is greater than the pre-defined k, which is the most likely scenario in practice, there are two options: (A) to terminate the process or (B) to forge ahead with brutal merging or eliminating.

   Option A: Coming to this point, the clusters in θ_1 are more dissimilar from each than the given threshold α_1. If the value of α_1 has reached the minimum acceptable similarity, it indicates that the clusters in θ_1 for the given dataset are too dissimilar from each other to be merged to obtain the intended k number of clusters. We then conclude that the pre-set value for k is unreasonable and unachievable, and output the generated clusters.



(a) The critical difference diagram of the first experiment.

(b) The critical difference diagram of the second experiment.

Fig. 3 The Critical difference diagram of the critical level of 0.1 in which it shows the comparison of six ensemble methods using eight datasets. **a** The critical difference diagram of the first experiment. **b** The critical difference diagram of the second experiment

   Option B: However, as there is no gold-standard for setting up the minimum acceptable similarity threshold, it is then also reasonable to go ahead with the process by adapting the threshold value α_1 to reflect the similarity distribution in the current similarity matrix S_c, and then merging the clusters with the above described step, or eliminating the clusters with the following steps. The elimination is carried out based on the cluster certainty. The certainty of each cluster in θ_1 is calculated by Eq. 7 and their certainty values are ranked in a descending order.

   i. If each of the top k clusters contains at least one certain object based on the current value of α_2, then these clusters are taken as the final candidate clusters. For the remaining clusters, they will be brutally "eliminated" by moving them from θ_1 to a new matrix θ_2 in order to be used in the next stage. The cluster similarity S_c and membership similarity S_x will be updated accordingly and α_1 will be adapted. Then, we move onto stage 3.

   ii. Otherwise, we adapt α_2 to be the maximum membership similarity to the *kth* cluster and consider the first k clusters as the final candidate clusters.

(c) When $\lambda < k$, i.e. the number of clusters in θ_1 is less than the pre-chosen k, then we consider if any clusters in θ_1 can be divided by adapting the value of α_1. In this case, it is possible that α_1 is unreasonably low and should be adapted incrementally to an appropriate value, then we should go back the beginning of this stage until the number of the clusters in θ_1 reaches k and move onto the next stage.

4.2.3 Stage 3: enforce hard clustering

The aim here is to ensure that each object is assigned to only one cluster. So, the inputs of this stage are: S_{x}, which is the membership similarity matrix; θ_2, which contains the membership similarity of the eliminated.

As defined earlier, for an object, if its maximum membership value $S_x(x_i, c_j) <= \alpha_2\,(\forall j = 1, \ldots, k)$, it is considered as an *uncertain object*, and as a *totally uncertain object* if its maximum membership value is zero. Four main steps are required as follows:

1. **Check whether θ_1 contains any totally uncertain objects.**
   There is a possibility that the previous stage may have resulted in totally uncertain objects in θ_1. This is of a particular concern during the elimination process, as this may have caused information to be lost for some objects, so we verify that each object in θ_1 has a membership value associated with at least one cluster. If θ_1 contains some totally uncertain objects, we calculate their neighbourhood similarity with clusters in θ_2. We are in fact modifying our early definition of neighbourhood similarity [1], by calculating the average occurrence of their objects' neighbours and the other objects placed in the candidate clusters. In other words, we calculate the similarity between the totally uncertain object and the candidate clusters in θ_1 as the average of how many times they are classified in the same cluster in θ_2 with other objects that are already placed in the candidate clusters in θ_1.
2. **Identify totally certain and certain objects in θ_1 as in definitions 5 and3.**
   As certain objects have a higher similarity value than α_2, we assign them to the cluster that has a maximum membership similarity among other clusters in θ_1.
3. **Measure the quality of each candidate cluster in θ_1.** In principle, any cluster quality measure can be used, so in this study we measure the compactness of the certain objects in a cluster as the quality metric, and here we call it the original quality of each cluster.
   The compactness of a cluster is usually measured by the variance, *Var*, which is the average of the squared differences from the mean, as follows:

$$Var(c) = \frac{1}{|\overleftarrow{c}|} \sum_{i=1}^{|\overleftarrow{c}|} (S_x(x_i, \overleftarrow{c}) - p_{\overleftarrow{c}})^2 \qquad (11)$$

It is basically the absolute value of the difference between the membership similarity value of object x_i in cluster $\overleftarrow{c}$, and the mean of the objects similarity in cluster $\overleftarrow{c}$ (cluster certainty $p_{\overleftarrow{c}}$ calculated by Eq. 7).

At the beginning, the size of each candidate cluster equals the total number of classified objects, and these objects are the only ones that we can assign to a candidate cluster with certainty, as they have the maximum membership similarity with the classified candidate clusters.

4. **Identify uncertain objects in θ_1 as in equation 6.**
   For each uncertain object the following steps are performed:

   (a) Identify the clusters of the current uncertain object in θ_1
   (b) For each identified cluster, we recalculate its quality using the Eq. 11 by including the current object membership similarity with the identified cluster.
   (c) Compare the original quality and the current quality of the identified clusters.
   (d) Assign the current object to the cluster that has a minimum effect on its original quality.
   (e) Increase the size of the assigned cluster by 1.
   (f) Update the original quality of the assigned cluster to be equal to the current quality.
   (g) Repeat steps until all the uncertain objects are assigned.

   Generally, we assign uncertain objects to a cluster in such a way that this will have a minimum effect on the latter's quality. By doing so, we aim to ensure that the original quality of the cluster has not been affected too much, as it is widely known that a small value for cluster quality indicates a compact cluster result.

Therefore, by assigning each object to only one cluster we obtain the final clustering result P^* of dataset X. A simple example for illustrating how the ACE works is given in "Appendix".

Algorithm 1: The Pseudocode for the second stage of the ACE Algorithm.

```
Input  : $\Gamma = \{P_1, P_2, P_3, \ldots, P_m\}$, $\alpha_1$, $\alpha_2$,
         $\alpha_{1min}$, $\Delta\alpha$, and $k$
Output: $P^*$
$\theta_1 \leftarrow$ Transform $m$ members into binary vectors
of initial clusters;
$S_c \leftarrow$ Compute cluster similarity $S_c$ for clusters
in $\theta_1$ with equation 1;
while true do
    $\theta_1 \leftarrow$ MergeCls(initial clusters, $S_c$, $\alpha_1$);
    if # clusters in $\theta_1, \lambda >= k$ then
        break;
    else
        Adapt $\alpha_1 = \alpha_1 + \Delta\alpha$;
$\lambda \leftarrow$ find # of clusters in $\theta_1$;
while $\lambda >= k$ do
    Update $S_c$ with equation 1;
    Adapt $\alpha_1 \leftarrow$ maximum similarity value in $S_c$;
    if $\alpha_1 < \alpha_{1min}$ then
        break;
    else
        $new\theta_1 \leftarrow$ MergeCls($\theta_1$, $S_c$, $\alpha_1$);
    if # clusters in $new\theta_1 < k$ then
        break;
    else
        $\theta_1 \leftarrow new\theta_1$
Compute similarity measure $S_x$ with equation 3;
$nb_{cls} \leftarrow$ find # clusters in $S_x$ that contain at
least one certain object specified by $\alpha_2$;
if $nb_{cls} == k$ then
    Consider these cluster as candidate clusters
    in $P^*$;
    $\theta_2 \leftarrow$ non-candidate clusters;
else
    Compute cluster certainty in $S_x$ with
    equation 8;
    Sort the cluster certainties in descend order;
    Adapt $\alpha_2 \leftarrow S_x max\{k\}$;
    Keep the top $k$ clusters in $S_x$ as the
    candiadate clusters;
    Remove the remaining clusters in $S_x$ to $\theta_2$;
$P^* \leftarrow$ AssignObjectToOnlyOneCluster($S_x$,
$\theta_2, \alpha_2$);
```

5 Experiments

We test the effectiveness of the ACE algorithm using eight real-world datasets from the UCI Machine Leaning Repository [32]: Iris, Wine, Thyroid, Multiple Features (Mfeatures), Glass, Breast Cancer Wisconsin (Bcw), Soybean and Ionosphere dataset. Table 1 shows the details of these datasets. Bcw has an attribute with missing values in some objects, which is removed. We also removed the second attribute in the Ionosphere dataset as only a single value (0) was present.

Two experiments were designed. In the first experiment, we set the number of clusters k equal to the true number of classes for each dataset which is fixed for all generated members, whereas in the second experiment we set a different number of clusters k for each member chosen randomly from the interval $[k-2, k+2]$. We chose this interval because we already know the number of clusters in the tested datasets so the minimum of this interval is set to less than k by 2 and the maximum set to a value larger than k by 2.

In both experiments, we set $\alpha 1 = 0.8, \alpha_2 = 0.7, \alpha_{1min} = 0.6$, and $\Delta\alpha = 0.1$, and we followed the common clustering ensemble framework as shown in Fig. 1. In the generation phase, we implemented the same techniques used by Ren et al. [33] in order to generate 10 diverse members. Thus, we used *k-means* to generate 5 members with a random sampling of 70% of the data, and we calculated the Euclidean distance between the remaining objects and the cluster centres and assigned them to the closest cluster. For each of the remaining members we ran *k-means* on 70% of randomly selected features.

The main aim of these experiments is to test the performance of ACE in these two particular situations. Also to see how effective our algorithms are compared to other competitive clustering ensemble algorithms, which include Co-association using the Average linkage method [11], ONCE also using the Average linkage method [1], DSCE [2], DICLENS [29] and MCLA [36]. We ran the algorithm 10 times, and each time the performance was measured by ARI and NMI.

When *ARI* and *NMI* are applied to evaluate the clustering results, one of the clustering partitions should be the ground “ true” partition of the data, P^t which, in practice, is normally assumed to be the class labels as there are no other true answers that can be used to verify the quality (accuracy) of the clustering result. The other partition is the clustering result of the ensemble that needs to be evaluated P^*.

ARI is the corrected version of the Rand index (RI), and is defined as follows:

Table 1 Details of the datasets used in experiments

Dataset	# Objects	# Features	# Clusters
Iris	150	4	3
Wine	178	13	3
Thyroid	215	5	3
Mfeatures	2000	2	10
Glass	214	9	6
Bcw	683	9	2
Soybean	47	35	4
Ionosphere	351	34	2

$$ARI(P^*, P^t) = \frac{RI(P^*, P^t) - Expected[RI]}{1 - Expected[RI]} \tag{12}$$

Where RI is calculated by:

$$RI(P^*, P^t) = \frac{n_{11} + n_{00}}{n_{00} + n_{11} + n_{10} + n_{01}} \tag{13}$$

Where n_{11} denotes the number of object pairs assigned to the same clusters in both P^* and P^t. n_{00} denotes the number of object pairs assigned to different clusters in P^* and P^t. n_{10} denotes the number of object pairs assigned to the same cluster in P^* and to different clusters in P^t. n_{01} denotes the number of object pairs assigned to different clusters P^* and to the same cluster in P^t. With simple algebra, the Adjust Rand Index [18] can be simplified to:

$$ARI = \frac{\sum_{i=1}^{k}\sum_{j=1}^{k}\binom{n_{ij}}{2} - [\sum_{i=1}^{k}\binom{n_i}{2}\sum_{j=1}^{k}\binom{n_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_{i=1}^{k}\binom{n_i}{2} + \sum_{j=1}^{k}\binom{n_j}{2}] - [\sum_{i=1}^{k}\binom{n_i}{2}\sum_{j=1}^{k}\binom{n_j}{2}]/\binom{n}{2}} \tag{14}$$

where n is the total number of objects in X, n_{ij} is the number of objects in the intersection of clusters $c_i \in P^*$ and $c_j \in P^t$, n_i and n_j are the numbers of objects in clusters $c_i \in P^t$ and $c_j \in P^*$ respectively, and $\binom{n}{2}$ is the binomial coefficient $\frac{n!}{2!(n-2)!}$. The maximum value of ARI is equal to 1, which means that P^* is identical to P^t, and it has an expected value 0 for independent clusterings. It is not necessarily for the number of clusters in P^* and P^t to be the same [25].

NMI is computed according to the average mutual information between every pair of clusters and class labels. Consider P^* the final clustering result of the ensemble and the ground-truth clustering P^t for dataset X. The NMI of the two partitions is defined as follows:

$$NMI(P^*, P^t) = \frac{\sum_{i=1}^{k^*}\sum_{j=1}^{k^t} n_{ij}\log(\frac{n.n_{ij}}{n_i.n_j})}{\sqrt{(\sum_{i=1}^{k^*} n_i \log(\frac{n_i}{n}))(\sum_{j=1}^{k^t} n_j \log(\frac{n_j}{n}))}} \tag{15}$$

The maximum value of NMI is equal to 1, which means that P^* is identical to P^t and the minimum value is equal to 0, when P^* is completely different from P^t. i.e. $n_{ij} = 0$, $(\forall i = \{1, \dots, k^*\}$, and $\forall j = \{1, \dots, k^t\})$.

6 Results and analysis

6.1 Results of ensembles built with fixed *k*

Tables 2 and 3 show the average value of ten runs of the compared algorithms measured by ARI and NMI respectively, along with their corresponding standard deviations. The bold value in each row shows the best performance in each dataset in terms of the quality of the clustering result and the underlined number shows the best value in terms of consistency. The last column of Table 2 and 3 represent the average performance of the generated members, and the last two rows show the average accuracy for each ensemble method over all datasets, as well as the average consistency of each method.

6.1.1 Results of ARI index

There are a number of interesting observations. Firstly, the performance of ACE is better than CO-average and ONCE-average in five datasets, whereas it performed very closely to them on other datasets. In particular, in the Iris, Thyroid and Glass datasets, ACE produced the highest results: 0.734, 0.611 and 0.534 respectively. Secondly, we noticed that ACE achieved the same performance as CO, DSCE and MCLA algorithms in the Bcw dataset, and that is the highest accurate result for this dataset. Thirdly, we noticed that ACE outperformed DICLENS in all datasets except in the Soybeans dataset, and we will explain later this particular situation for DICLENS. However, on average the DSCE algorithm achieved the best performance compared with other algorithms, followed closely by the ACE algorithm.

In terms of consistency measured by the standard deviation, ACE was the most consistent algorithm in the thyroid dataset compared with the others, and it achieved a very close value to the most consistent algorithm in the most examined datasets such as the Bcw, Mfeatures and Wine datasets. The worst performance for the ACE algorithm was on the Soybean dataset, where it achieved a value equal to 0.081 compared with other algorithms, but this is still a small value.

Looking at the average performance of the generated members, we found that all the ensemble methods outperformed the average of members in all the datasets, except DICLENS which performed lower than the average members in the Glass and Mfeatures datasets as well as ACE in the Soybeans dataset.

However, the ACE algorithm performed second-best on average compared with the others, and it is close to the best performing algorithm measured by the ARI index, which is DSCE under these experimental settings.

6.1.2 Results of NMI index

In summary, these results are very similar to the results represented by ARI explained in the previous paragraph. The only difference is that on average the ACE achieved the best performance, along with the DSCE algorithm, measured by NMI.

Under this experimental set-up, i.e. with a fixed value for k for each dataset, ACE does not show a superiority to its predecessor DSCE, although it does in comparison to the other methods. However, it is worth noting that its predecessor DSCE has an obvious weakness, which is that it can only work with fixed k values, which limits its application on real-world problems when the true number of clusters, k, is not known in advance. That is why we extended DSCE to ACE to cope with variable numbers of clusters generated by the members. The next experiment is designed to demonstrate and compare their capability.

6.2 Results of ensembles built with random variable *k*

We did not run the DSCE algorithm in this experimental set-up as it is not capable of dealing with variable numbers of clusters generated by the members in an ensemble. All the other methods were run for comparison.

6.2.1 Results of ARI index

Table 4 shows the average performance measured by the ARI index along with the standard deviation in each dataset, and the average performance of the generated members. The results indicate that the ACE algorithm mostly performs better than the investigated collection of clustering ensemble algorithms. This is particularly true in five datasets, which are Wine, Glass, Bcw, Soybean and Ionosphere, whereas in Iris, Thyroid and Mfeatures it achieved a result close to the

Table 2 First experiment results: The average performance and the standard deviation of ten runs for each dataset measured by ARI, including the average performance (Ave-P) of each ensemble method across 8 datasets as well as the average consistency (Ave-Con)

	CO-Average	ONCE-Average	DSCE	ACE	DICLENS	MCLA	Ave-mem
Iris	0.725 ± 0.012	0.726 ± 0.009	0.732 ± 0.021	**0.734** ± 0.023	0.680 ± 0.077	0.723 ± 0.012	0.702 ± 0.038
Wine	0.369 ± 0.005	0.369 ± 0.005	**0.377** ± 0.025	0.371 ± 0.008	0.369 ± 0.005	0.372 ± 0.002	0.366 ± 0.004
Thyroid	0.559 ± 0.024	0.584 ± 0.044	0.609 ± 0.032	**0.613** ± 0.023	0.582 ± 0.044	0.563 ± 0.025	0.473 ± 0.036
Mfeatures	0.315 ± 0.006	**0.316** ± 0.005	**0.316** ± 0.004	0.314 ± 0.008	0.290 ± 0.069	0.308 ± 0.021	0.293 ± 0.029
Glass	0.509 ± 0.029	0.526 ± 0.030	0.528 ± 0.027	**0.535** ± 0.029	0.392 ± 0.123	0.534±0.020	0.501 ± 0.009
Bcw	**0.849** ± 0.004	0.847 ± 0.003	**0.849** ± 0.004	**0.849** ± 0.004	0.842 ± 0.005	**0.849** ±0.004	0.830 ± 0.021
Soybean	0.547 ± 0.006	0.550 ± 0.015	0.578 ± 0.052	0.532 ± 0.081	**0.632** ± 0.046	0.548 ± 0.006	0.566 ± 0.025
Ionosphere	0.163 ± 0.014	0.166 ± 0.008	**0.169** ± 0.005	0.165 ± 0.008	0.161 ± 0.009	0.166 ± 0.006	0.149 ± 0.007
Ave-P	0.505	0.511	**0.520**	0.514	0.493	0.508	0.443
Ave-C	0.012	0.015	0.017	0.023	0.048	0.012	0.031

The bold value in each row shows the best performance in each dataset in terms of the quality of the clustering result and the underlined number shows the best value in terms of consistency

Table 3 First experiment results: The average performance and the standard deviation of ten runs for each dataset measured by NMI Index, including the average performance (Ave-P) of each ensemble method across 8 datasets as well as the average consistency (Ave-Con)

	CO-Average	ONCE-Average	DSCE	ACE	DICLENS	MCLA	Ave-mem
Iris	0.751 ± 0.015	0.752 ± 0.012	0.763 ± 0.024	**0.766** ± 0.028	0.757 ± 0.008	0.749 ± 0.015	0.737 ± 0.025
Wine	0.428 ± 0.003	0.428 ± 0.003	**0.432** ± 0.014	0.429 ± 0.006	0.427 ± 0.004	0.429 ± 0.001	0.428 ± 0.003
Thyroid	0.434 ± 0.047	0.473 ± 0.062	0.480 ± 0.056	**0.531** ± 0.042	0.501 ± 0.053	0.418 ± 0.033	0.403 ± 0.026
Mfeatures	**0.479** ± 0.002	**0.479** ± 0.003	**0.479** ± 0.002	0.478 ± 0.007	0.468 ± 0.026	0.475 ± 0.009	0.460 ± 0.027
Glass	0.712 ± 0.027	0.725 ± 0.029	0.725 ± 0.021	0.726 ± 0.022	0.617 ± 0.107	**0.728** ± 0.017	0.704 ± 0.007
Bcw	0.750 ± 0.005	0.749 ± 0.004	0.750 ± 0.005	**0.751** ± 0.005	0.742 ± 0.006	**0.751** ± 0.005	0.731 ± 0.023
Soybean	0.717 ± 0.002	0.723 ± 0.024	0.756 ± 0.064	0.712 ± 0.076	**0.822** ± 0.056	0.717 ± 0.002	0.736 ± 0.019
Ionosphere	0.122 ± 0.014	0.124 ± 0.009	**0.128** ± 0.005	0.123 ± 0.008	0.119 ± 0.009	0.124 ± 0.006	0.108 ± 0.006
Ave-P	0.549	0.557	**0.564**	**0.564**	0.557	0.549	
Ave-C	0.015	0.018	0.024	0.024	0.034	0.011	

The bold value in each row shows the best performance in each dataset in terms of the quality of the clustering result and the underlined number shows the best value in terms of consistency highest performance in these datasets, which are ONCE in Mfeatures and MCLA in the other two datasets. However, the result on the Mfeatures dataset indicates that ACE is applicable to a large dataset.

The ACE also enhances the performance of the generated members in all of the investigated datasets except the Ionosphere dataset, which is slightly better than the clustering ensemble algorithms; this may be due to random *k* in these members.

In terms of consistency, ACE was more consistent in two datasets, which are Glass and Bcw, while in the Iris, Wine and Ionosphere datasets it was the second most consistent algorithm compared with other algorithms. On average, three algorithms achieved very close results in terms of consistency; these are MCLA, ONCE and ACE, which are equal to 0.035, 0.037 and 0.038 respectively.

6.2.2 Results of NMI index

Similar experimental results are also observed using the NMI index shown in Table 5, where ACE achieved the highest performance on three datasets: Iris, Bcw, and Ionosphere. However, in Wine, Mfeatures and Glass it achieved very close results to the highest performance. In the Soybean dataset the highest performance was achieved by the DICLENS algorithm, which also performed very well in the Wine and Mfeatures datasets. These results were only achieved by the NMI and not the ARI index, which leads us to investigate further the number of clusters discovered by DICLENS, as it has the ability to discover *k* automatically. This is in contrast to other examined clustering ensemble algorithms, in which *k* is provided by the user in advance.

6.2.3 Identifying the true number of clusters in DICLENS

Figure 4 shows the number of clusters discovered by the DICLENS algorithm in all tested datasets over ten runs. It is observed that the number of clusters in most datasets is unstable and changeable over the ten runs. This has an effect on the NMI index, which is an information theory based index that measures the shared information between two clustering results. Most of the DICLENS results in the majority of datasets have fewer clusters than the actual true labels in the data.

This means that when we compare the produced clusters with the actual clusters, it is clear that the produced clusters share more objects with the actual clusters, as each produced cluster can share with more than one cluster in the true label and that can lead the NMI result to be increased.

For example, it was highlighted in the Wine dataset over the ten runs that the discovered *k* was equal to 2 which is less than the number of the true labels, 3. Therefore, the NMI measure, as it is based on how much information the compared clustering results share, unfairly indicates that this result is more accurate than ACE. Moreover, in the Soybean dataset the discovered *k* is equal to 2 in three runs, 3 in four runs and 4 in the remaining three runs, whereas the number of the true labels is equal to 4. It is obvious that fewer clusters shared more objects with more true clusters in this case, and the NMI scored higher than ARI compared with other clustering results obtained by other algorithms. It is observed that when the number of clusters in the compared results is less than the number of true labels of the data, the NMI measure inappropriately indicates that this result is more accurate than others that have produced exactly the number of the true clusters.

Another important investigation is on the subject of relations between the performance of the experimented clustering ensemble methods and the two types of ensembles being explored. It has been demonstrated that on average the ACE is more accurate than CO, ONCE, DICLES and MCLA algorithms, across the two types of ensemble examined.

Table 4 Second experiment results: the average performance and the standard deviation of ten runs for each dataset measured by ARI, including the average performance of each ensemble method across 8 datasets

	CO-Average	ONCE-Average	ACE	DICLENS	MCLA	Ave-mem
Iris	0.669 ± 0.065	0.674 ± 0.057	0.696 ± 0.038	0.565 ± 0.009	**0.722** ± 0.043	0.605 ± 0.029
Wine	0.324 ± 0.045	0.344 ± 0.060	**0.403** ± 0.014	0.367 ± 0.024	0.393 ± 0.008	0.326 ± 0.011
Thyroid	0.252 ± 0.175	0.189 ± 0.121	0.303 ± 0.032	0.308 ± 0.118	**0.448** ± 0.119	0.285 ± 0.053
Mfeatures	0.325 ± 0.002	**0.326** ± 0.001	0.325 ± 0.005	0.324 ± 0.006	0.277 ± 0.013	0.321 ± 0.005
Glass	0.265 ± 0.006	0.259 ± 0.008	**0.269** ± 0.004	0.200 ± 0.048	0.152 ± 0.022	0.258 ± 0.005
Bcw	0.866 ± 0.018	0.860 ± 0.016	**0.869** ± 0.014	0.853 ± 0.031	0.864 ± 0.014	0.773 ± 0.037
Soybean	0.534 ± 0.000	0.534 ± 0.000	**0.578** ± 0.160	0.575 ± 0.070	0.547 ± 0.039	0.547 ± 0.036
Ionosphere	0.076 ± 0.047	0.037 ± 0.035	**0.084** ± 0.034	0.076 ± 0.039	0.061 ± 0.019	0.117 ± 0.014
Ave-P	0.414	0.403	**0.441**	0.409	0.433	0.404
Ave-C	0.045	0.037	0.038	0.043	**0.035**	0.024

The bold value in each row shows the best performance in each dataset in terms of the quality of the clustering result and the underlined number shows the best value in terms of consistency

Table 5 Second experiment results: the average performance and the standard deviation of ten runs for each dataset measured by NMI, including the average performance of each ensemble method across 8 datasets

	CO-Average	ONCE-Average	ACE	DICLENS	MCLA	Ave-mem
Iris	0.753 ± 0.017	0.749 ± 0.027	**0.766** ± 0.032	0.753 ± 0.026	0.755 ± 0.037	0.706 ± 0.012
Wine	0.406 ± 0.010	0.415 ± 0.022	0.421 ± 0.014	**0.435** ± 0.018	0.415 ± 0.005	0.410 ± 0.010
Thyroid	0.293 ± 0.077	0.250 ± 0.066	0.308 ± 0.050	0.331 ± 0.040	**0.356** ± 0.048	0.302 ± 0.035
Mfeatures	0.486 ± 0.002	0.487 ± 0.002	0.490 ± 0.008	**0.493** ± 0.005	0.464 ± 0.007	0.484 ± 0.005
Glass	0.441 ± 0.018	**0.449** ± 0.016	0.430 ± 0.016	0.389 ± 0.032	0.307 ± 0.032	0.423 ± 0.011
Bcw	0.773 ± 0.024	0.765 ± 0.021	**0.776** ± 0.019	0.759 ± 0.032	0.770 ± 0.019	0.687 ± 0.028
Soybeans	0.710 ± 0.000	0.710 ± 0.000	0.722 ± 0.127	**0.767** ± 0.070	0.716 ± 0.018	0.734 ± 0.020
Ionosphere	0.043 ± 0.035	0.023 ± 0.012	**0.048** ± 0.026	0.043 ± 0.029	0.030 ± 0.013	0.099 ± 0.016
Ave-P	0.488	0.481	0.495	**0.496**	0.477	0.480
Ave-C	0.023	**0.021**	0.036	0.032	0.022	0.017

The bold value in each row shows the best performance in each dataset in terms of the quality of the clustering result and the underlined number shows the best value in terms of consistency

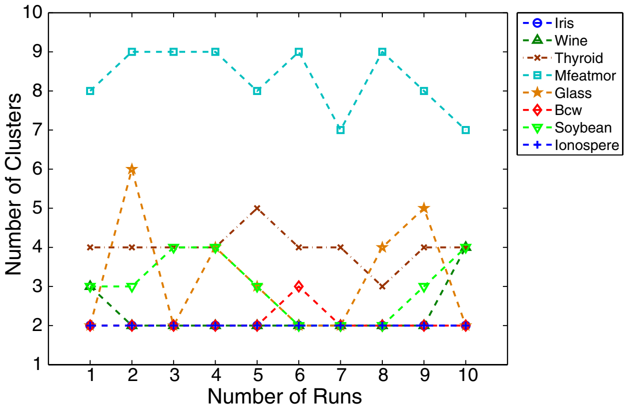


Fig. 4 Number of clusters produced by DICLENS algorithm for each dataset in ten runs for the result in the second experiment

6.3 Test of improvement significance

We tested the statistical significance of the results of the two experiments that we have done in Sects. 6.1 and 6.2 on the two types of ensembles.

We applied the Iman-Davenport test [22] to the results in Table 2 and Table 4 under the null hypothesis that the mean ranks are equal for all the examined algorithms. The significant level is set to 0.1 by default. For the first experiment, we can reject the null hypothesis of the mean rank of the performance being equal for all algorithms (the Iman-Davenport test result is equal to 4.4051 which gives a small p-value equal to 0.0032, which indicates that there is a significant difference). For the second experiment in Table 4, the Iman-Davenport test result is equal to 2.5434, which gave a small p-value equal to 0.0617, indicating that there is a significant difference.

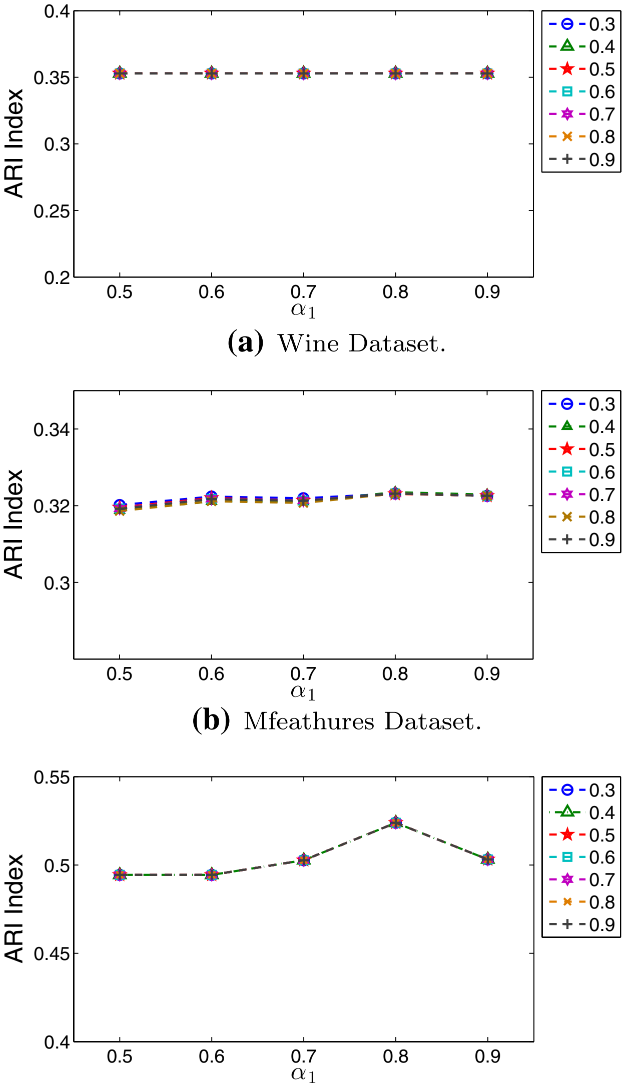
Therefore, we proceeded with the Nemenyi test as a post-hoc test for a pairwise comparison to discover where the differences lie. Figure 5a shows the critical difference diagram of the critical level of 0.1 for the results presented in Table 2, and the critical difference CD is equal to 2.4218. As we can see from the diagram, we have two solid bars which show two groups of algorithms in cliques, indicating that there is no statistically significant difference between algorithms in the same group, whereas there is a significant difference between algorithms in the different groups. We observed that, based on the average ranks, DSCE was first followed by ACE and then MCLA. Moreover, DICLENS was last in this average ranking. This demonstrated that there is a significant difference between ACE and DICLENS and CO algorithms, but not between ACE and DSCE, MCLA and ONCE based on this experimental set-up.

Figure 5b shows the critical difference diagram of the results presented in Table 4. As we can see, there are two groups of algorithms in two cliques. The first group includes ACE, MCLA, CO and DICLENS, whereas the second group includes MCLA, CO, DICLENS and ONCE. The results indicate that there is a significant difference between algorithms placed in different groups and in this case between the ACE and ONCE algorithms in this experimental set-up, although ACE is ranked the first with a considerable distance from the second algorithm MCLA.

7 Analysis of parameters and time complexity

There are two parameters in *ACE* which are α_1 and α_2. α_1, as stated previously, is the minimum similarity allowed between initial clusters, whereas α_2 is the relative membership certainty threshold for classifying objects in candidate clusters.

To find out how these parameters can affect the quality of the final clustering result of the ACE, we analyse them with



(a) Wine Dataset.

(b) Mfeathures Dataset.

(c) Glass Dataset.

Fig. 5 The Average of ARI index of ten runs for analysing the two parameters α_1 and α_2 using members with fixed k **a** Wine Dataset, **b** Mfeathures Dataset and **c** Glass Dataset

the two types of ensembles as we did in Sect. 5, using Wine, Mfeatures and Glass datasets as an illustration. For the second type of ensemble, we allow for α_1 to take a higher value than its value in the first experiment, due to the fact that when the members have different k from one another they are more dissimilar than when they have fixed k. Therefore α_1 can take a value between 0.5 and 0.9 in the first experiment, whereas in the second experiment it takes a value between 0.3 and 0.9.

However, in the first experiment, we ran ACE with a different initial values of α_1, and each one of them with all the possible values for α_2 ten times. We firstly ran the *k-means* algorithm to generate 10 members all with the fixed k equal to the true number of classes for each dataset.

Figure 6 illustrates the effect of different values of α_1 and α_2 on the average performance of the ensemble built by members with a fixed k, this average is for ten runs measured by the ARI index. We noticed that on the Wine dataset the average performance of ADCE is the same for all values of α_1 and α_2; this indicates that the ACE is not sensitive to its parameters.

In the Mfeatures dataset, the average performance of ACE is slightly improved when α_1 is equal to 0.8 and 0.9. It is noticed that all the values of α_2 have the same performance with all the values of α_1. The average performance of ACE in the Glass dataset is the same when α_1 is equal to 0.5 and 0.6, which is slightly improved when α_1 is equal to 0.7 and 0.9; when it is equal to 0.8 it reaches its highest performance.

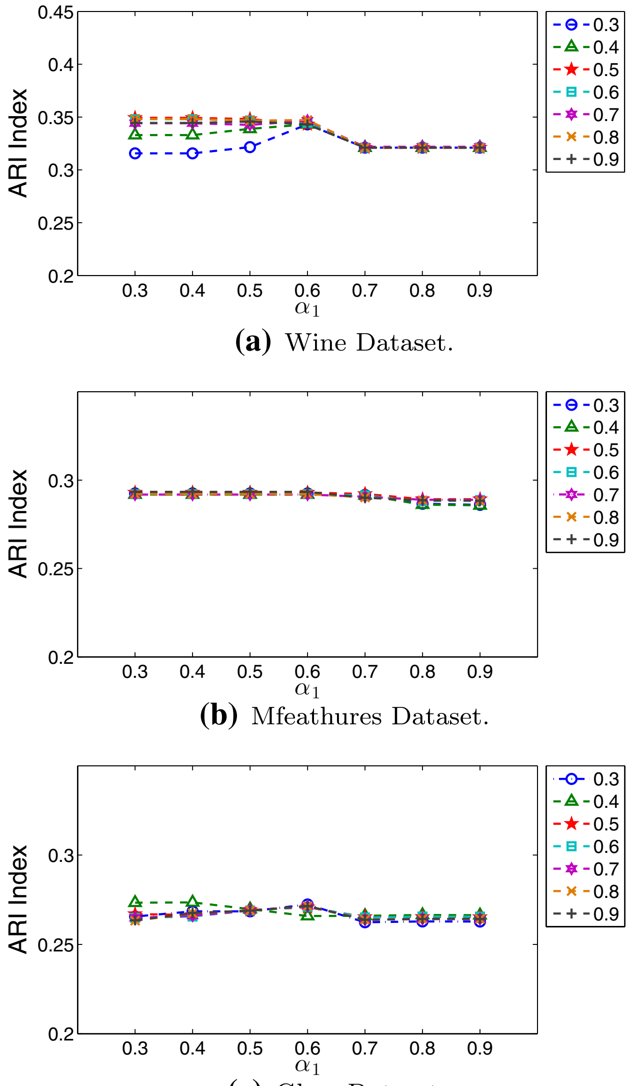
It is noticed that all values of α_2 achieved the same performance with all values of α_1 in all the examined datasets, indicating that the different values of α_2 have no effect on the performance of the ACE when it is built with members that have fixed k.

On the other hand, Fig. 7 illustrates the effect of the different values of two parameters on the average performance of the ACE ensemble built with members having a random variable k. We can see that in the Wine dataset the ACE performance is decreased a little when α_1 is equal to 0.7 in which the performance remains stable with 0.8 and 0.9 in all possible values of α_2. In the Mfeatures dataset, the ACE performance is slightly improved when α_1 is less than 0.7. However, in the Glass dataset the ACE performance fluctuates with a slight increase to reach a value of 0.6 and then a slight drop when α_1 is equal to 0.7 after a stable performance. It is noticed that with all the possible values of α_2 that the average performance of ACE remains the same in almost all cases for α_1. Therefore, the results suggested that α_2 has no effect on the performance of ACE, and α_1 has a slight effect on ACE performance. A value between 0.6 and 0.8 is better for an ensemble built with fixed k, whereas a value between 0.3 and 0.5 is better for an ensemble built with different k and when α_2 is between 0.5 to 0.9, as these values have no effect on the ACE performance.

The time complexity for the worst-case scenario of ACE algorithm is estimated to be equal to $O(k_m^2(k_m + n_u))$, where k_m is the total number of clusters in all the generated members, and n_u is the number of uncertain objects which is in the worst case scenario equal to $(n_u = n - k)$, and k is the number of pre-defined clusters for the dataset.

8 Conclusion

In this paper, we have proposed a new clustering ensemble algorithm, named the Adaptive Dual-Similarity Clustering Ensemble, which is capable of dealing with different and variable numbers of clusters generated by the members of the same dataset. The novelty is the consensus function that measures the similarity between the clusters themselves, and between clusters and their assigned objects. It works in three stages. The first is the transformation stage, where the initial clusters are transformed into binary vector representation, and the second stage calculates the similarity between initial clusters; this captures the relationship between clusters and merges the most similar clusters to produce the intended k consensus clusters. The final stage identifies the object's certainty of being assigned in the initial clusters. It focuses on the cluster quality and resolves the uncertain objects by assigning them to a cluster in a way that has a minimum effect on its quality. We tested our proposed method on eight real-world benchmark datasets. The results show that on average our proposed ensemble algorithm outperforms the state-of-the-art cluster ensemble algorithms, which include the MCLA, CO, ONCE and DICLENS algorithms. There are a number of advantages to the proposed method; firstly, it avoids relabelling problems when aggregating multiple clustering results. Secondly, it utilises the information of similarity between clusters and membership of objects to clusters to generate consensus clusters. Thirdly, it is able to deal with the clustering members that have different numbers of clusters and convert them exactly or very closely to the true number of clusters in the final clustering result. Finally and more noticeably, it is more efficient–instead of calculating the similarity between objects like others do, it calculates the similarity between the initial clusters of the ensemble members, which is much smaller than the number of objects, hence the proposed method has the potential to be applied in big data clustering problems.



(a) Wine Dataset.

(b) Mfeathures Dataset.

(c) Glass Dataset.

Fig. 6 The Average of ARI index of ten runs for analysing the two parameters α_1 and α_2 using members with random k **a** Wine Dataset, **b** Mfeathures Dataset and **c** Glass Dataset

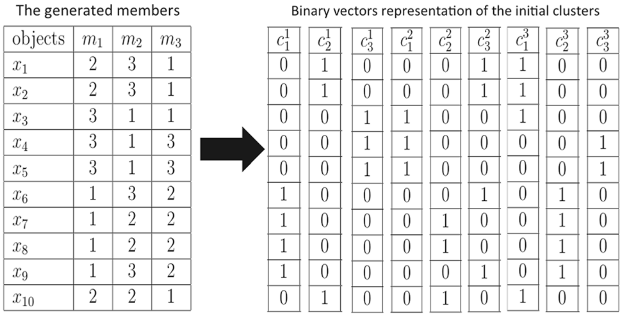
Further work should include investigating the different factors affecting the ensemble performance, such as member diversity and quality. This might help to identify a suitable selective strategy to be incorporated into ACE in order to improve its performance further.

Appendix A: An illustrating example for the ACE

We illustrate how the ACE works with a simple example. Suppose we have a dataset X that contains 10 objects, $X = \{x_1, x_2, \dots, x_{10}\}$ and that we have generated 3 members ($m = 3$), each of which has 3 clusters ($k = 3$). We set $\alpha_1 = 0.8$, $\alpha_2 = 0.5$, and $k = 3$, and we run the ACE algorithm in three stages as follows:

Stage 1: Transformation. We transform the members into a binary vector representation as shown in Figure 7, in which each cluster in the generated member is represented by a binary vector with 9 binary vectors in total. For example, vector c_3^2 is the third cluster in the second member m_2. Four objects x_1, x_2, x_6, x_9 were assigned to cluster c_3^2, so we set their value equal to 1, whereas for other objects in c_3^2 we set a value of 0. These vectors are the input of the second stage.

Stage2: Generating Consensus Cluster Stage. In this stage, we first measure the similarity between the initial



The generated members

objects	m_1	m_2	m_3
x_1	2	3	1
x_2	2	3	1
x_3	3	1	1
x_4	3	1	3
x_5	3	1	3
x_6	1	3	2
x_7	1	2	2
x_8	1	2	2
x_9	1	3	2
x_{10}	2	2	1

Binary vectors representation of the initial clusters

c_1^1	c_2^1	c_3^1	c_1^2	c_2^2	c_3^2	c_1^3	c_2^3	c_3^3
0	1	0	0	0	1	1	0	0
0	1	0	0	0	1	1	0	0
0	0	1	1	0	0	1	0	0
0	0	1	1	0	0	0	0	1
0	0	1	1	0	0	0	0	1
1	0	0	0	0	1	0	1	0
1	0	0	0	1	0	0	1	0
1	0	0	0	1	0	0	1	0
1	0	0	0	0	1	0	1	0
0	1	0	0	1	0	1	0	0

Fig. 7 An illustrative example of three clustering members for dataset X of 10 objects, and the transformation from members into a binary matrix representation

clusters, and generate the similarity matrix S_c as shown in Table 6. Then we perform the merging process as follows:

Firstly, we set α_1 equal to 0.8. Looking at S_c, we find that c_1^1 and c_2^3 are identical and have a similarity greater than α_1 with c_2^2, so we merge them by replacing them with $\overleftarrow{c}_1$, which contains the summation of their object membership. In addition, c_2^1 have a similarity greater than α_1 with c_3^2 and c_1^3, so we merge them too as $\overleftarrow{c}_2$. We also merge c_3^1 and c_1^2 as $\overleftarrow{c}_3$. As a result, we gain four clusters, $\overleftarrow{c}_1$, $\overleftarrow{c}_2$, $\overleftarrow{c}_3$ and $\overleftarrow{c}_4$ in the updated θ_1, as shown in Table 7. Then we recalculate the similarity measures S_c for the updated θ_1 as shown in Table 8.

Based on α_1, we find that there are no more similar clusters to be merged in the updated similarity matrix S_c. For the third step in stage 2, we first check the number of clusters (λ) in θ_1, and we find that $\lambda = 4$, which is larger than k. Then we apply Option B by measuring the cluster similarity S_c for clusters in θ_1 as shown in Table 9 and we adapt α_2 to the maximum similarity in S_c, which is equal to 0.764. We merge $\overleftarrow{c}_3$ and $\overleftarrow{c}_4$ and we updated θ_1 as shown in Table 12. As a result we obtain $\lambda = k = 3$. Then we calculate the membership similarity S_x as shown in Table 10, and we move to stage 3

Table 7 The result of θ_1 after we merge the most similar clusters, which are $\overleftarrow{c}_1 = \{c_1^1 + c_2^2 + c_2^3\}$, $\overleftarrow{c}_2 = \{c_2^1 + c_3^2 + c_1^3\}$, $\overleftarrow{c}_3 = \{c_3^1 + c_1^2\}$ and $\overleftarrow{c}_4 = \{c_3^3\}$

	$\overleftarrow{c}_1$	$\overleftarrow{c}_2$	$\overleftarrow{c}_3$	$\overleftarrow{c}_4$
x_1	0	3	0	0
x_2	0	3	0	0
x_3	0	1	2	0
x_4	0	0	2	1
x_5	0	0	2	1
x_6	2	1	0	0
x_7	3	0	0	0
x_8	3	0	0	0
x_9	3	0	0	0
x_{10}	0	3	0	0

Table 8 The updated Similarity Matrix S_c after the first step of the merging process is performed, which is the result of measuring the similarity between four clusters in θ_1 (in Table 7)

	$\overleftarrow{c}_1$	$\overleftarrow{c}_2$	$\overleftarrow{c}_3$	$\overleftarrow{c}_4$
$\overleftarrow{c}_1$	–	−0.408	−0.535	−0.408
$\overleftarrow{c}_2$	−0.408	–	−0.218	−0.500
$\overleftarrow{c}_3$	−0.535	−0.218	–	0.764
$\overleftarrow{c}_4$	−0.408	−0.500	0.764	–

Stage 3: Enforce Hard Clustering. This stage start by first identifying totally certain and certain objects. So, based on α_2, we identify $x_1, x_2, x_4, x_5, x_7, x_8, x_9$ and x_{10} as totally certain objects, while we identify x_3 and x_6 as certain objects. Then we assign them to the candidate cluster that has a maximum membership similarity among other candidates, and S_x is updated as shown in table 11.

Then we check whether S_x contains any uncertain objects and it does not, so we produce the final clustering result $P^* = \{2, 2, 3, 3, 3, 1, 1, 1, 1, 2\}$.

Table 6 The Similarity Matrix S_c, which is the result of measuring the similarity between initials cluster vectors in our illustrative example (Figure 7) using S_c measure. $- - -$ cells indicates that this similarity is not calculated as they are placed in the same member

	c_1^1	c_2^1	c_3^1	c_1^2	c_2^2	c_3^2	c_1^3	c_2^3	c_3^3
c_1^1	–	–	–	−0.535	0.802	−0.250	−0.667	1	−0.408
c_2^1	–	–	–	−0.429	−0.429	0.802	0.802	−0.535	−0.327
c_3^1	–	–	–	1	−0.429	−0.535	−0.089	−0.535	0.764
c_1^2	−0.535	−0.429	1	–	–	–	−0.089	−0.535	0.764
c_2^2	0.802	−0.429	−0.429	–	–	–	−0.535	0.802	−0.327
c_3^2	−0.250	0.802	−0.535	–	–	–	0.583	−0.250	−0.408
c_1^3	−0.667	0.802	−0.089	−0.089	−0.535	0.583	–	–	–
c_2^3	1	−0.535	−0.535	−0.535	0.802	−0.250	–	–	–
c_3^3	−0.408	−0.327	0.764	0.764	−0.327	−0.408	–	–	–

Table 9 The result of updating θ_1 after we merge $\overleftarrow{c}_3$ and $\overleftarrow{c}_4$ by summing their objects membership similarity and result in $\overleftarrow{c}_3$

	$\overleftarrow{c}_1$	$\overleftarrow{c}_2$	$\overleftarrow{c}_3$
x_1	0	3	0
x_2	0	3	0
x_3	0	1	2
x_4	0	0	3
x_5	0	0	3
x_6	2	1	0
x_7	3	0	0
x_8	3	0	0
x_9	3	0	0
x_{10}	0	3	0

Table 10 The results of S_x after no more merging step is needed

	$\overleftarrow{c}_1$	$\overleftarrow{c}_2$	$\overleftarrow{c}_3$
x_1	0	1	0
x_2	0	1	0
x_3	0	0.33	0.67
x_4	0	0	1
x_5	0	0	1
x_6	0.67	0.33	0
x_7	1	0	0
x_8	1	0	0
x_9	1	0	0
x_{10}	0	1	0

Table 11 The results of assigning totally certain and certain objects to the candidate cluster

	$\overleftarrow{c}_1$	$\overleftarrow{c}_2$	$\overleftarrow{c}_3$
x_1	0	1	0
x_2	0	1	0
x_3	0	0	0.6
x_4	0	0	1
x_5	0	0	1
x_6	0.6	0	0
x_7	1	0	0
x_8	1	0	0
x_9	1	0	0
x_{10}	0	1	0

Table 12 The result of S_x after we perform the second stage

	$\overleftarrow{c}_1$	$\overleftarrow{c}_2$	$\overleftarrow{c}_3$
x_1	0	1	0
x_2	0	1	0
x_3	0	0.3	0.6
x_4	0	0	0.6
x_5	0	0	0.6
x_6	0.6	0.3	0
x_7	1	0	0
x_8	1	0	0
x_9	1	0	0
x_{10}	0	1	0

Assume that we set $\alpha_2 = 0.9$, which is a high value. The number of clusters (nb_{cls}) in S_x that contain at least one certain object is equal to 2. As there is no further merging process to be done, we calculate S_x, which is shown in Table 11. Then we implement the elimination process that is described in Option B (steps *i* to *iv*). So, for each cluster in S_x, we calculate their certainties (using equation 8), and we obtain $\rho_{\overleftarrow{c_1}} = 0.9$, $\rho_{\overleftarrow{c_2}} = 0.85$, $\rho_{\overleftarrow{c_3}} = 0.6$, $\rho_{\overleftarrow{c_4}} = 0.3$. We rank these certainties in descending order and we obtain $\{0.9, 0.72, 0.6, 0.3\}$. Then we adapt α_2 to the maximum certainties of the *kth* clusters in this ranked list, which is equal 0.6. As result, we identify $\overleftarrow{c_1}, \overleftarrow{c_2}$ and $\overleftarrow{c_3}$ as candidate clusters and we eliminate $\overleftarrow{c_4}$. We update S_x accordingly as shown in Table 12.

Then we move onto stage 3, and based on α_1 we identify x_1, x_2, x_7, x_8, x_9 and x_{10} as totally certain objects, and we identify other objects as uncertain objects. We measure the quality of the candidate clusters using equation 12 as follows:

$$Var(\overleftarrow{c_1}) = \frac{1}{3}((1-0.9)^2 + (1-0.9)^2 + (1-0.9)^2) = 0.01$$

$$Var(\overleftarrow{c_2}) = \frac{1}{3}((1-0.72)^2 + (1-0.72)^2 + (1-0.72)^2) = 0.0784$$

$$Var(\overleftarrow{c_3}) = 0$$

Then, we iterate on uncertain objects, and we proceed with steps (*a*) to (*e*). The detailed results of these steps for object x_3 are as follows:

(a) For each candidate cluster we recalculate its quality by including this time x_3:

$$Var(\overleftarrow{c_1}) = \frac{1}{4}((1-0.9)^2 + (1-0.9)^2 + (1-0.9)^2 + (0-0.9)^2) = 0.21$$

$$Var(\overleftarrow{c_2}) = \frac{1}{4}((1-0.72)^2 + (1-0.72)^2 + (1-0.72)^2 + (0.3-0.72^2)) = 0.1029$$

$$Var(\overleftarrow{c_3}) = \frac{1}{1}((0.6-0.6)^2) = 0$$

(b) We compare for each cluster the original quality and the current quality:

$$Var(\overleftarrow{c_1}) = 0.21 - 0.01 = 0.2,$$

$$Var(\overleftarrow{c_2}) = 0.1029 - 0.0784 = 0.0245,$$

$$Var(\overleftarrow{c_3}) = 0 - 0 = 0$$

(c) We assign x_3 to the cluster that has a minimum effect on its quality, that is done as follows: $min\{0.2, 0.0245, 0\} = 0$. So, we assign x_3 to cluster $\overleftarrow{c_3}$.

(d) We increase the size of $\overleftarrow{c_3}$ by 1.

(e) We update the original quality of $\overleftarrow{c_3}$ to be equal to the current quality.

After all the uncertain objects are assigned, we produce the final clustering result, which is : $P^* = \{2, 2, 3, 3, 3, 1, 1, 1, 1, 2\}$.

References

1. Alqurashi T, Wang W (2014) Object-neighbourhood clustering ensemble method. In: International conference on intelligent data engineering and automated learning (IDEAL). Springer, Spain, pp 142–149
2. Alqurashi T, Wang W (2015) A new consensus function based on dual-similarity measurements for clustering ensemble. In: International conference of data science and advanced analytics (DSAA). IEEE/ACM, pp 149–155
3. Ayad HG, Kamel MS (2005) Cluster-based cumulative ensembles. Multiple Classifier Systems. Springer, New York, pp 236–245
4. Ayad HG, Kamel MS (2010) On voting-based consensus of cluster ensembles. Pattern Recogn 43(5):1943–1953
5. Blumer A, Ehrenfeucht A, Haussler D, Warmuth MK (1987) Occam's razor. Inf Process Lett 24(6):377–380
6. Dimitriadou E, Weingessel A, Hornik K (2002) A combination scheme for fuzzy clustering. Int J Pattern Recogn Artif Intell 16(07):901–912
7. Dudoit S, Fridlyand J (2003) Bagging to improve the accuracy of a clustering procedure. Bioinformatics 19(9):1090–1099
8. Fern XZ, Brodley CE (2003) Random projection for high dimensional data clustering: a cluster ensemble approach. In: Proceedings of the 20th international conference on machine learning, pp 186–193. http://www.aaai.org/Papers/ICML/2003/ICML03-027.pdf. Accessed 10 Mar 2014
9. Fern XZ, Brodley CE (2004) Solving cluster ensemble problems by bipartite graph partitioning. In: Proceedings of the 21st International Conference on Machine learning. ACM, New York, p 36
10. Fred AL, Jain AK (2002) Data clustering using evidence accumulation. In: Proceedings of the16th International Conference on Pattern Recognition, vol 4. IEEE, pp 276–280
11. Fred AL, Jain AK (2005) Combining multiple clusterings using evidence accumulation. IEEE Trans Pattern Anal Mach Intell 27(6):835–850
12. Gionis A, Mannila H, Tsaparas P (2007) Clustering aggregation. ACM Trans Knowl Discov Data (TKDD) 1(1):4
13. Han J, Kamber M, Pei J (2006) Data mining: Concepts and techniques. Morgan Kaufmann, Burlington
14. Hore P, Hall LO, Goldgof DB (2009) A scalable framework for cluster ensembles. Pattern Recogn 42(5):676–688
15. Houle ME (2008) The relevant-set correlation model for data clustering. Stat Anal Data Mining 1(3):157–176
16. Huang D, Lai J, Wang CD (2016a) Ensemble clustering using factor graph. Pattern Recogn 50:131–142
17. Huang D, Lai J, Wang CD (2016b) Robust ensemble clustering using probability trajectories. IEEE Trans Knowl Data Eng 28:1312–1326
18. Hubert L, Arabie P (1985) Comparing partitions. J Classif 2(1):193–218
19. Iam-on N, Boongoen T, Garrett S (2010) LCE: a link-based cluster ensemble method for improved gene expression data analysis. Bioinformatics 26(12):1513–1519
20. Iam-On N, Boongoen T, Garrett S, Price C (2011) A link-based approach to the cluster ensemble problem. IEEE Trans Pattern Anal Mach Intell 33(12):2396–2409
21. Iam-On N, Boongeon T, Garrett S, Price C (2012) A link-based cluster ensemble approach for categorical data clustering. IEEE Trans Knowl Data Eng 24(3):413–425
22. Iman RL, Davenport JM (1980) Approximations of the critical region of the fbietkan statistic. Commun Stat Theory Methods 9(6):571–595
23. Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv (CSUR) 31(3):264–323
24. Karypis G, Kumar V (1998) A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J Sci Comput 20(1):359–392
25. Kuncheva LI, Hadjitodorov ST (2004) Using diversity in cluster ensembles. In: Proceedings of the IEEE international conference on systems, man and cybernetics, vol 2, pp 1214–1219
26. Li T, Ding C, Jordan M et al (2007) Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization. In: Proceedings of the IEEE International Conference on Data Mining (ICDM). IEEE, pp 577–582
27. Li Z, Wu XM, Chang SF (2012) Segmentation using superpixels: a bipartite graph partitioning approach. In: The IEEE conference on computer vision and pattern recognition (CVPR), pp 789–796
28. Luo H, Jing F, Xie X (2006) Combining multiple clusterings using information theory based genetic algorithm. In: Proceedings of the International Conference on Computational Intelligence and Security, vol 1. IEEE, pp 84–89
29. Mimaroglu S, Aksehirli E (2012) DICLENS: divisive clustering ensemble with automatic cluster number. IEEE/ACM Trans Comput Biol Bioinform (TCBB) 9(2):408–420
30. Minaei-Bidgoli B, Topchy A, Punch WF (2004) Ensembles of partitions via data resampling. In: Proceedings of the International Conference on Information Technology: coding and computing ITCC, vol 2. IEEE, pp 188–192
31. Monti S, Tamayo P, Mesirov J, Golub T (2003) Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. Mach Learn 52(1–2):91–118
32. Moshe L (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml. Accessed 2 Oct 2013
33. Ren Y, Zhang G, Domeniconi C, Yu G (2013) Weighted-object ensemble clustering. In: Proceedings of the IEEE 13th International Conference on Data Mining (ICDM). IEEE, pp 627–636
34. Sevillano X, Socoró JC, Alıas F (2009) Fuzzy clusterers combination by positional voting for robust document clustering. Proc del lenguaje Nat 43:245–253
35. Strehl A, Ghosh J (2000) Value-based customer grouping from large retail data sets. In: AeroSense, International Society for Optics and Photonics, pp 33–42
36. Strehl A, Ghosh J (2003) Cluster ensembles–a knowledge reuse framework for multiple partitions. J Mach Learn Res 3:583–617
37. Swift S, Tucker A, Vinciotti V, Martin N, Orengo C, Liu X, Kellam P (2004) Consensus clustering and functional interpretation of gene-expression data. Genome Biol 5(11):R94
38. Tan PN, Steinbach M, Kumar V (2006) Introduction to data mining. Pearson Addison Wesley, Boston
39. Topchy A, Jain AK, Punch W (2004) A mixture model of clustering ensembles. In: Proceedings of the SIAM International Conference of Data Mining. Citeseer

40. Topchy A, Jain AK, Punch W (2005) Clustering ensembles: models of consensus and weak partitions. IEEE Trans Pattern Anal Mach Intell 27(12):1866–1881
41. Vega-Pons S, Ruiz-Shulcloper J (2011) A survey of clustering ensemble algorithms. Int J Pattern Recogn Artif Intell 25(03):337–372
42. Vega-Pons S, Correa-Morris J, Ruiz-Shulcloper J (2010) Weighted partition consensus via kernels. Pattern Recogn 43(8):2712–2724
43. Vinh NX, Houle ME (2010) A set correlation model for partitional clustering. In: Advances in Knowledge Discovery and Data Mining. Springer, New York, pp 4–15
44. Wang W (2008) Some fundamental issues in ensemble methods. In: Proceedings of the IEEE international joint conference on neural networks, pp 2243–2250
45. Yi J, Yang T, Jin R, Jain AK, Mahdavi M (2012) Robust ensemble clustering by matrix completion. In: Proceedings of the IEEE 12th International Conference on Data Mining (ICDM). IEEE, pp 1176–1181
46. Yoon HS, Ahn SY, Lee SH, Cho SB, Kim JH (2006) Heterogeneous clustering ensemble method for combining different cluster results. In: Data Mining for Biomedical Applications. Springer, New York, pp 82–92
47. Zhou ZH, Tang W (2006) Clusterer ensemble. Knowl-Based Syst 19(1):77–83