CA-Tree: A Hierarchical Structure for Efficient and Scalable Co-association Based Cluster Ensembles

Tsaipei Wang, member, IEEE
Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
E-mail: wangts@cs.nctu.edu.tw

Abstract

Cluster ensembles have attracted a lot of research interest in recent years, and their applications continue to expand. Among the various algorithms for cluster ensembles, those based on co-association matrices are probably the ones studied and used the most because co-association matrices are easy to understand and implement. However, the main limitation of co-association matrices as the data structure for combining multiple clusterings is the complexity that is at least quadratic to $N$, the number of patterns. In this paper we propose CA-tree, a dendogram-like hierarchical data structure, to facilitate efficient and scalable cluster ensembles for co-association matrix based algorithms. All the properties of the CA-tree are derived from base clustering labels and do not require the access to the original data features. We then apply a threshold to the CA-tree to obtain a set of nodes, which are then used in place of the original patterns for ensemble clustering algorithms. The experiments demonstrate that the complexity for co-association based cluster ensembles can be reduced to close to linear to $N$ with minimal loss on clustering accuracy.
I. Overview

1.1 Introduction

Clustering is the process of identifying the underlying groups or structures in a set of patterns without the use of class labels. While there have been a large set of clustering algorithms (see [1]-[3] for reviews), they all have their limitations in terms of data characteristics that can be processed and types of clusters that can be found. The performance of many clustering algorithms also strongly depends on proper choices of parameters and/or initializations. As a result, the choice of appropriate clustering algorithms and/or parameters is highly problem dependent, and often involves lots of heuristic choices or trial-and-error.

Following the success of classifier ensembles [4]-[6], cluster ensembles, the process of obtaining a combined clustering of the data using the information from multiple clusterings, have attracted much research interest in the past several years. The motivation behind cluster ensembles is that it is more likely to generate more stable, more reliable, or more meaningful clustering results with less sensitivity to parameter choices or initializations by combining multiple different clusterings of the same data. According to [7], ensembles of clusterings have advantages over individual clusterings in several aspects: robustness (better average performance), novelty (new solution different from the individual clusterings), stability and confidence estimation, and parallelism and scalability (combination of clusterings obtained at different time, locations, or with different features). Ensemble clustering is identified in [8] as one of three major frontiers of clustering techniques in recent years. Many existing experimental results have indicated improved clustering results compared to the results of single clustering runs.
1.2. Relevant Works in Cluster Ensembles

Algorithms for cluster ensembles consist of two main components. The first component is the method for generating the base clusterings (the individual clusterings in the ensemble), including the source of diversity (differences among the base clusterings). One possibility is to use different clustering algorithms for each base clustering. This is the approach in [9] where the goal is to identify "robust" clusters, each defined as a set of patterns that are in the same cluster in every base clustering. When a single algorithm is employed to produce all the base clusterings, the most common choices include expectation-maximization (EM), as in [10][11], and k-means, as in [12]-[14]. Both EM and k-means have the built-in source of diversity from different initializations and, as in [12], different numbers of clusters in the base clusterings. Other possible sources of diversity include different orderings of the patterns for on-line clustering algorithms [15] and different linkage types for hierarchical agglomeration [16]. Different subsampling of the data has been studied as well [17]. The relation between the degree of diversity and the quality of the combined clustering is the subject of [18].

A popular approach is to generate the base clusterings using different subspace projections of the original feature space [11], especially when there are more than just a few features. It is indicated in [11] that such clustering ensembles work better than dimensionality reduction techniques such as principal component analysis (PCA). Some have also employed a similar approach where a randomly selected subset of features is used in each base clustering [10][14]. An extreme example is to use a large number of random 1-D projections, which is studied in [19] to demonstrate the effectiveness of combining many weak clusterings.

The second major component is sometimes called the "consensus function" and is concerned about the integration and representation of the combined information from multiple
clusterings, and the extraction of a final clustering from this representation. An example is to use a prototype in the original feature space to represent a cluster in a base clustering [20]. However, this method is not applicable to the "knowledge-reuse framework", as suggested by [21], where the consensus function only has access to the base cluster labels of the patterns, but not to the original features.

A straightforward approach when using only the base cluster labels is to consider the cluster ensemble as a transform of each pattern into a "label space", where each feature is the cluster label of a pattern in a base clustering. Some early works employ a voting scheme to select the final cluster label of each pattern [22][23]. However, this requires solving the correspondence problem among the base clusterings through re-labeling, a process that is somewhat problematic especially when the base clusterings have different numbers of clusters. More recently, [24] describes cumulative voting methods that are able to handle different numbers of clusters in the base clusterings. One can also apply a clustering algorithm to the patterns in the label space. Examples include the use of k-modes [25], EM [7], and the optimization of quadratic mutual information [7].

Graph-based representations have also been popular for combining the information from multiple clusterings. Representative algorithms include Cluster-based Similarity Partitioning Algorithm (CSPA), Hyper-Graph Partitioning Algorithm (HGPA), and Meta-CLustering Algorithm (MCLA), all three proposed in [21], as well as Hybrid Bipartite Graph Formulation (HBGF) [26]. An appropriate graph-cut algorithm can then be applied to determine the final clustering according to the type of graph (patterns as vertices for CSPA, clusters as hyperedges for HGPA, clusters as vertices in a meta-graph for MCLA, and a pattern-cluster bipartite graph for HBGF).

7
The co-association matrix [12][27] is perhaps the most widely used data structure for combining the information from multiple clusterings. Example applications to practical problems include [28]-[31]. A co-association matrix is a square matrix where each element represents the similarity between two patterns. This similarity is given by the probability of these two patterns being in the same cluster among all the base clusterings. In a word, a co-association matrix provides a relational representation of a data set according to the base clusterings. [32] describes a method for building the co-association matrix for data sets with mixed numerical and categorical features.

The final clustering can be derived from the co-association matrix using many different clustering algorithms for relational data. The most common choices are probably hierarchical agglomeration with various linkage types [11][12][27][30]. Graph-cut algorithms are used in CSPA [21] as well as in [28], with the co-association matrix treated as a graph where each vertex represents a pattern and the weight of each edge represents the similarity between two patterns. Other examples include the use of spectral clustering in [32] and fuzzy k-means in [14], where each column is treated as a new transformed feature vector.

1.3. Motivations and Contributions

The main drawback of using co-association matrices in cluster ensembles is its complexity. Since a co-association matrix has $N^2$ elements ($N$ being the number of patterns), it has a memory complexity of $O(N^2)$, and the time complexity is also at least $O(N^2)$ for both its creation and its partition into the final clustering. This quadratic complexity makes co-association matrices unsuitable for large data sets. This problem seriously limits the applicability of related algorithms. For example, in [10] where several different algorithms for cluster ensembles are compared, CSPA, which uses co-association matrices, is only applied to data sets of no more than 2000
A straightforward approach to solve this complexity problem is to simply sub-sample the data. Slightly more robust is to first group the patterns into a large number of small pre-clusters (we use the term "pre-clusters" here to distinguish them from the clusters generated by the subsequent clustering algorithms) using a common algorithm like k-means. The prototypes of these pre-clusters are just their centroids in the feature space. The subsequent ensemble clustering is then applied to these prototypes instead of the original patterns, and the final cluster labels of the original patterns are determined from the final cluster labels of the prototypes according to the nearest-neighbor rule. This is the approach taken by [12], which also adds an additional step of outlier removal based on shared nearest-neighbor analysis. An example of applying ensemble clustering to 301 prototypes for a data set of 4000 patterns is given in [12], although there is no discussion regarding the effect of using prototypes on clustering results.

While the prototype-based method above for complexity reduction is intuitive, it has several limitations that affect its applicability. First, there is no general rule for selecting the number of prototypes. If the number of prototypes is just set to a fixed percentage of \( N \), which is the approach in [33], the complexity of using co-association matrices is still at least quadratic to \( N \). Second, the prototypes are points in the feature space. This means that we need to use all the features at once in the pre-clustering process. As a result, we are unable to use this method in multi-view clustering (multiple clusterings obtained using different subsets of features or projections to subspaces). In addition, it becomes a complicated, if not impossible, task to incorporate existing clustering results in the form of cluster labels, a scenario mentioned in the discussion of the knowledge-reuse framework in [21].

Our goal in this paper is to present a new method for reducing the time and memory
complexity of co-association based cluster ensemble algorithms, therefore significantly extending their applicability to larger data sets. A hierarchical structure called a co-association tree (CA-tree for short) similar to a dendogram is built using the base cluster labels. A cut of this "dendogram" at a given threshold gives a preliminary partition of the data set into disjoint groups similar to the pre-clusters. We then compute the co-association matrix and obtain the final clustering using the representatives of these groups. The name CA-tree arises from the fact that the size of a node is the minimum "degree of co-association" (defined in the same way as the elements of an ordinary co-association matrix) between the representative of this node and all its descendents. We list several advantages of CA-trees in the following:

- The procedure for building CA-trees utilizes only the cluster labels of the patterns in the base clusterings without the need to access the original features. Therefore, CA-trees are applicable to multi-view clustering and are able to incorporate past clustering results, therefore resolving the limitations of the prototype-based methods.

- Compared with prototype-based methods, we do not have the poorly defined task of determining the number of prototypes. Instead, we only need to select a threshold. In addition, our experimental results indicate that the same threshold (at a fixed ratio of the number of base clusterings) can work well across different data sets.

- We experimentally observe that the number of groups resulting from a cut to a CA-tree approximately follows a power-law relation with respect to \( N \), with the exponent less than one and often less than 0.5. Therefore, the complexity of co-association based ensembles using these groups, being quadratic to their numbers, is much lower than \( O(N^2) \) and often lower than \( O(N) \). For the latter case, the complexity with respect to \( N \) is \( O(N) \) from the procedure for building the CA-tree.
- We propose a node reduction scheme to further reduce the number of groups as well as the resulting complexity. We also point out potentially valuable information regarding the actual cluster characteristics (compact or elongated/curvilinear) by comparing the number of nodes before and after node reduction (see subsection 4.6).

1.4. Summary of Notations

<table>
<thead>
<tr>
<th>Data set and base clusterings</th>
<th>Properties of CA-tree nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$ number of patterns</td>
<td>$z$ a node in the CA-tree</td>
</tr>
<tr>
<td>$X = {x_i}$ set of patterns ($1 \leq i \leq N$)</td>
<td>$X(z)$ the set of data points associated with $z$</td>
</tr>
<tr>
<td>$H$ number of base clusterings</td>
<td>$G(z)$ the set of distinct $\lambda_i$ in $X(z)$</td>
</tr>
<tr>
<td>$P_h$ the $h$th base clustering ($1 \leq h \leq H$)</td>
<td>$\lambda(z)$ partial label vector shared by $z$ and all its descendents</td>
</tr>
<tr>
<td>$k_h$ number of clusters in $P_h$</td>
<td>$\lambda_h(z)$ representative label vector of $z$</td>
</tr>
<tr>
<td>$k_{\min} / k_{\max}$ lower and upper bounds of $k_h$</td>
<td>$D(z)$ &quot;radius&quot; of $z$ in the space of label vectors</td>
</tr>
<tr>
<td>$\lambda_{ih}$ base cluster label of $x_i$ in $P_h$</td>
<td>$D'(z)$ an estimated upper bound of $D(z)$</td>
</tr>
<tr>
<td>$\lambda_i$ label vector (the vector of all the $\lambda_{ih}$) of $x_i$</td>
<td>$Z(z)$ a subset of the descendents of $z$ used for computing $D'(z)$</td>
</tr>
<tr>
<td>$\Lambda$ set of all the distinct $\lambda_i$</td>
<td>$n_{des}$ a bound on the size of $Z(z)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node selection</th>
<th>Combined clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau$ threshold used to select a set of nodes</td>
<td>$S^*(\tau) = [s_{ij}(\tau)]$ co-association matrix obtained using $Z(\tau)$</td>
</tr>
<tr>
<td>$Z(\tau)$ the set of selected nodes at $\tau$</td>
<td>$Q(\tau)$ clustering accuracy at $\tau$</td>
</tr>
<tr>
<td>$N_Z(\tau)$ the number of nodes in $Z(\tau)$</td>
<td>$t_{total}$ total time for getting the final clustering from a dataset</td>
</tr>
<tr>
<td>$\gamma$ the ratio of retained patterns in node reduction</td>
<td>$t_{CE}$ time for getting the final clustering from base clusterings</td>
</tr>
<tr>
<td>$Z'(\tau)$ the version of $Z(\tau)$ without node reduction</td>
<td>$N_Z^*(\tau)$ the version of $N_Z(\tau)$ without node reduction</td>
</tr>
<tr>
<td>$\alpha(\tau)$ growth rate of $N_Z(\tau)$ relative to $N$</td>
<td>$\alpha'$ the maximum growth rate of $N_Z(\tau)$ relative to $N$</td>
</tr>
</tbody>
</table>

1.5. Paper Organization

For the rest of this paper, we first review the definition of co-association matrices in Section II. Definitions and algorithms related to our tree structure are covered in Section III. Section IV
presents the experimental results. We provide the analysis of complexity as well as experiments on computation time in Section V, followed by the conclusions in Section VI.

II. Co-association Matrices

Assume that \( X = \{x_1, x_2, \ldots, x_N\} \) is a data set containing \( N \) patterns. Let \( P = \{C_1, C_2, \ldots, C_k\} \) be a crisp clustering (partition) of \( X \). Here \( k \) is the number of clusters in \( P \). The clusters, \( C_1, C_2, \ldots, C_k \), are disjoint non-empty subsets of \( X \), with their union being \( X \). It is possible to obtain many different partitions of \( X \). Let a cluster ensemble consists of \( H \) clusterings of \( X \): \( P_1, P_2, \ldots, P_H \). These are called the base clusterings of the ensemble. We allow each base clustering to have a different number of clusters and use \( k_h \) to represent the number of clusters in \( P_h \) \((1 \leq h \leq H)\). We also define an \( H \)-element label vector for each \( x_i \) as

\[
\lambda_i = [\lambda_{i1}, \lambda_{i2}, \ldots, \lambda_{iH}]
\]

with its \( h \)th element, denoted as \( \lambda_{ih} \), being the cluster label of \( x_i \) in \( P_h \). We use the Hamming distance between two label vectors \( \lambda_i \) and \( \lambda_j \) (i.e., the number of different cluster labels) as their dissimilarity, denoted as \( d(\lambda_i, \lambda_j) \).

We define a \( NxN \) matrix, \( S^{(h)} = [s_{ij}^{(h)}] \), for each base clustering \( P_h \) according to

\[
s_{ij}^{(h)} = \begin{cases} 
1, & \lambda_{ih} = \lambda_{jh} \\
0, & \text{otherwise}
\end{cases}
\]

(2)

The overall co-association matrix of a cluster ensemble, denoted as \( S^* = [s_{ij}^*] \), is simply the average of all the \( S^{(h)} \):

\[
s_{ij}^* = \frac{1}{H} \sum_{1 \leq h \leq H} s_{ij}^{(h)}.
\]

(3)

An equivalent definition using the label vectors is
III. The CA-Tree

3.1. Core Groups

The first source of size reduction of a co-association matrix arises from the observation that often multiple patterns share the same label vector. A set of patterns with the same label vector are indistinguishable for the cluster ensemble algorithms. As they belong to the same cluster in each base clustering, we shall expect that they are assigned to the same cluster in the final clustering regardless of the actual cluster ensemble algorithm used. Therefore, these patterns can be treated as a single entry, resulting in a smaller co-association matrix and the reduction of associated computational time and memory requirement.

For this purpose, we introduce here the concept of core groups, defined as subsets of $X$ that satisfy the following condition: Two patterns $x_i$ and $x_j$ belong to the same core group if and only if $\lambda_i = \lambda_j$. Let $A$ be the set of all the different label vectors in $X$ given the base clusterings. Since each core group has a unique label vector, the number of core groups is simply $|A|$, the cardinality of $A$. It is easy to understand that $|A| \leq N$. The number of core groups depends on the distribution of patterns as well as the algorithms and parameters used to generate the base clusterings. The procedure for identifying the core groups from the base clusterings is included in the next subsection. We note here that the concept of core groups has been previously mentioned in [16], although the method there for identifying the core groups is very different from ours and is only designed to work with a particular cluster ensemble algorithm.

The following example demonstrates the use of a $|A| \times |A|$ co-association matrix of the core groups in place of a $N \times N$ co-association matrix of the original patterns. Using a 2-D synthetic
data set of 4 Gaussian clusters of 25 points each (Fig. 1(a)), four base clusterings are generated using k-means with \( k \) being 3, 4, 5, and 6, respectively. The base clusterings are shown in Fig. 1(d)-(g), with the numbers in the plots being the cluster labels. These four base clusterings result in \(|\mathbf{A}|=9\) core groups as in Fig. 1(b), where the numbers represent the indices of the core groups. Here we see that a problem size of \(100^2\) in terms of the number of elements in the co-association matrix is reduced to \(9^2\) simply by taking advantage of the redundancy in base cluster labels. We list in Table I the associated label vector as well as the number of patterns of each core group. Fig. 1(c) plots the resulting 9x9 co-association matrix according to (3).

3.2. CA-Tree Construction

Further reduction of computational complexity can result from this assumption: Patterns with similar label vectors are more likely to be assigned to the same cluster in the final clustering. This leads to the use of groups that contain several similar core groups, instead of the individual core groups, as the units for building the co-association matrix. Such an approach results in even fewer groups than \(|\mathbf{A}|\) and further reduces the amount of computation and memory requirement.

This leaves us with the problem of how to form these groups from the core groups. Since our purpose is to lower the computational complexity below the quadratic complexity of the original co-association matrix, we specifically exclude any option that starts with a matrix of pairwise similarity or dissimilarity among the original patterns or the core groups. Instead, with each base clustering of the data, we incrementally grow a tree structure that has some similarity to a dendogram formed with hierarchical clustering algorithms. When all the base clusterings have been processed, a threshold is applied to the tree to extract a set of nodes, each representing a group of core groups. We can then build a co-association matrix of these groups as the input for final clustering extraction.
Each node of the tree contains one or more core groups of $X$. For a given node $z$, we define $X(z)$ as the union of these core groups (i.e., the elements of $X(z)$ are original patterns) and $G(z)$ as the set of label vectors associated with these core groups. We initialize the tree with a single root node where $X(z)=X$. For each additional base clustering $P$, if $X(z)$ of a leaf node $z$ belongs to more than one cluster in $P$, we add a child to $z$ for each different cluster label of $X(z)$ in $P$. The purpose is to ensure that $X(z)$ of a leaf node $z$ is always a core group. This process is described in the following pseudocode:

![Fig. 1. (a) A 4-cluster data set. (b) The 9 four core groups. Numbers in the plot indicate core group indices. (c) The co-association matrix of the core groups. (d)-(g) The 4 base clusterings used to obtain the core groups.](image)

<table>
<thead>
<tr>
<th>Core group Index</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label Vector</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Number of Patterns</td>
<td>12</td>
<td>13</td>
<td>2</td>
<td>23</td>
<td>24</td>
<td>1</td>
<td>10</td>
<td>3</td>
<td>12</td>
</tr>
</tbody>
</table>
All the processing within the outermost loop can be implemented with a single scan through all the patterns in $X$. As a result, the complexity of this algorithm is $O(NH)$, not including the generation of the base clusterings. As the algorithm terminates, each $\mathcal{A}(z)$ is the partial label vector that is shared among $z$ and all its descendents. If $z$ is a leaf node, $X(z)$ is a single core group and $G(z)$ contains a single label vector which is $\mathcal{A}(z)$. The largest possible branching factor in the tree is $k_{h_{\text{max}}} = \max(k_h)$ ($1 \leq h \leq H$) and the largest possible depth is $H$. We note here that the exact tree generated depends on the ordering of the clusterings. However, we do not expect the variation of clustering results among different orderings to be more significant than the variation among different ensembles generated under the same conditions. An example tree from the experiment for Fig. 1 is given in Fig. 2(a). The leaf nodes (nodes that correspond to the core groups) are all at the bottom. The vertical position of each non-leaf node indicates the index of
the base clustering (index $h$ in the pseudocode) at which its children are created.

### 3.3. Determining Node Sizes and Representatives

If we can define the "size" (in the space of label vectors) of a node $z$ to represent the consistency of the label vectors in $G(z)$, we can redraw the tree more like a dendogram with the vertical axis being the node size. Fig. 2(b) is such an example (the computational detail of node sizes is described later). By applying a threshold on the node size, we cut the tree at a particular level. The horizontal dashed line in Fig. 2(b) is just an example of a threshold that results in a 5-group partition; each of the 5 groups here corresponds to a node $z$ (marked by arrows in Fig. 2(b)) such that the size of $z$ is no more than the threshold while the size of $\text{parent}(z)$ is above the threshold. Some relevant definitions for this purpose are given below.

First, we define the distance between a label vector $\lambda$ and a node $z$ as the largest distance between $\lambda$ and all the label vectors in $G(z)$:

$$d(\lambda, z) = \max_{\lambda' \in G(z)} d(\lambda, \lambda').$$  \hspace{1cm} (5)
For each node \( z \), the label vector \( \lambda \) in \( G(z) \) that minimizes \( d(\lambda, z) \) is selected as the node representative \( \lambda_R(z) \):

\[
\lambda_R(z) = \arg \min_{\lambda \in G(z)} d(\lambda, z). \tag{6}
\]

Next we define the size of a node \( z \) in the space of cluster labels as

\[
D_r(z) = d(\lambda_R(z), z). \tag{7}
\]

The subscript 'r' indicates that its meaning is similar to radius. However, one problem with this definition is that its computation involves all the possible label vectors in \( G(z) \). For those nodes that are several levels above the leaf nodes, their \( G(z) \) may contain many label vectors, making the computation of (5) to (7) quadratic to \( \| G(z) \| \) and hence too time consuming. This is inconsistent with our overall goal to reduce computational complexity. As a result, we opt for estimating the upper bound of \( D_r(z) \) in place of the exact value. From here on, we use \( D_r'(z) \) to represent the estimated upper bound of \( D_r(z) \). The definitions (5) to (7) are replaced with the following estimations based on the estimated \( D_r'(z) \). First, we need to select a subset \( Z(z) \) of the descendents of \( z \) (the actual procedure of selecting \( Z(z) \) is described later). Equation (5) is replaced with

\[
d'(\lambda, z) = \min \left\{ H, \max_{z' \in Z'(z)} \left[ d(\lambda, \lambda_R(z')) + D_r'(z') \right] \right\}. \tag{8}
\]

The node representative is now determined as

\[
\lambda_R(z) = \arg \min_{\lambda \in G'(z)} d'(\lambda, z), \tag{9}
\]

where we consider only the node representatives of those nodes in \( Z(z) \),

\[
G'(z) = \left\{ \lambda_R(z') \mid z' \in Z'(z) \right\}. \tag{10}
\]

The size of \( z \) is now estimated as
\[ D_r'(z) = d'(\lambda_R(z), z). \] (11)

If \( z \) is a leaf node, we simply set \( D_r'(z) = 0 \) and \( \lambda_R(z) = \lambda(z) \). The actual computation of (8) to (11) for all the nodes is done in a bottom-up order; this ensures that the processing of a node occurs only after all its descendent nodes are already processed. During this process, we can also determine \( G(z) \) and \( X(z) \) according to

\[ G(z) = \bigcup_{z', \text{parent}(z') = z} G(z') \] (12)

and

\[ X(z) = \bigcup_{z', \text{parent}(z') = z} X(z'). \] (13)

Now let us explain the procedure for determining \( Z'(z) \). The purpose of \( Z'(z) \) is to reduce the computational complexity for node size and representative determination from proportional to \( ||G(z)||^2 \) to proportional to \( ||Z(z)||^2 \) per node. Here we need a parameter \( n_{des} \) to control \( ||Z(z)|| \). We start with \( Z'(z) \) containing only the immediate children of \( z \) and iteratively replace the largest node in \( Z'(z) \) with its immediate children. This process is continued until \( ||Z'(z)|| \geq n_{des} \) or all the nodes in \( Z'(z) \) are leaf nodes. Overall, \( ||Z(z)|| \) satisfies the condition that

\[ ||Z'(z)|| \leq \min \left( ||G(z)||, k_{h(\text{max})} + n_{des} \right). \] (14)

The actual value of \( n_{des} \) used in our experiments is 32 unless specified otherwise. The complexity of this step is therefore the number of nodes times the upper bound of \( ||Z'(z)||^2 \), or \( O(N \cdot H \cdot (k_{h(\text{max})} + n_{des})^2) \) as there are no more than \( 2N \) nodes. The factor \( H \) results from the need to compute the Hamming distances between label vectors.

The extraction of a set of nodes from the tree is very similar to the process of extracting a partition from the dendogram in hierarchical clustering algorithms. We determine \( Z(\tau) \), the extracted set of nodes given a threshold \( \tau \), according to
\[ Z(\tau) = \left\{ z \mid D_r(z) \leq \tau \text{ and } D_r(\text{parent}(z)) > \tau \right\}. \] (15)

It is those nodes in \( Z(\tau) \) that are used to build the co-association matrix. We also define \( N_Z(\tau) = |Z(\tau)| \). Let us index the nodes in \( Z(\tau) \) as \( z_1(\tau), z_2(\tau), \ldots, z_{N_Z(\tau)}(\tau) \). A straightforward method for computing the elements of the resulting co-association matrix, denoted as \( S^*(\tau) \), is just to use the pairwise similarities among the node representatives in a form similar to (4):

\[ s^*_{ij}(\tau) = 1 - \frac{1}{H} d[\lambda_R(z_i(\tau)), \lambda_R(z_j(\tau))]. \] (16)

In Fig. 3(a)-(d) we show the partitions of \( X \) obtained by thresholding the tree in Fig. 2(b) with \( \tau \) being 0, 1, 2, and 3, respectively. The values of \( N_Z(\tau) \) is 9, 5, 4, and 3, respectively. It is interesting to see that the partition in Fig. 3(c) is completely consistent with the ground truth.

### 3.4. Node Reduction

One interesting observation in Fig. 3(a) is that several of the 9 core groups contain very few patterns. These core groups are generally located at low-density regions or regions between actual clusters in the feature space. On the other hand, core groups in high-density regions are more likely to contain more patterns. Actually, here the 6 largest core groups out of 9 contain 94% of the patterns. Similar phenomena also occur for the nodes extracted at different \( \tau \) (that is,
not just the core groups) after more experiments with larger data sets; the results are included in
the next section.

Based on this observation, we believe it is possible to further reduce \( N_Z(\tau) \) by keeping only
the important nodes (i.e., nodes that contain substantial numbers of patterns) for building the
coa-association matrix. Instead of specifying the number of nodes to keep, we define a parameter
\( \gamma (0 < \gamma \leq 1) \) such that we retain enough nodes, in the order of decreasing number of patterns, to
include at least a total of \( \gamma N \) patterns. For example, for the data in Fig. 3, using \( \gamma = 0.8 \) results in
\( N_Z(\tau) \) of 5, 4, 4, and 3 for \( \tau \) of 0, 1, 2, and 3, respectively. The choice of \( \gamma \) in a particular
application depends on the desired tradeoff between clustering accuracy and efficiency; we can
always err on a larger \( \gamma \) to prevent significant degradation of the final cluster accuracy. The value
0.9 (in addition to the default, 1.0) is used in some experiments in the next section and seems to
work well for all our data sets. Since now both \( Z(\tau) \) and \( N_Z(\tau) \) are also affected by \( \gamma \), we use \( Z^*(\tau) \)
and \( N_Z^*(\tau) \) to represent the versions without node reduction (i.e., when \( \gamma = 1 \)) when there might
be ambiguity.

As the final clustering is now derived from the retained nodes, a problem remains regarding
how to assign the final cluster labels for the patterns in the excluded nodes. We first extract a
subtree consisting of only the retained nodes and their ancestors. For each excluded node \( z \), we
perform a search starting at the root of this subtree. In each step, we select the child that is most
similar to \( z \). This search continues until we reach a leaf node, which is a retained node included
in the final clustering. The final cluster label for this leaf node is then assigned to \( z \) as well. The
complexity of this step is \( O(N_Z^*(\tau) \cdot H \cdot k_{bh(max)}) \). On the other hand, the complexity of selecting the
nodes to retain is \( O(N_Z^*(\tau) \cdot \log N_Z^*(\tau)) \) as this involves sorting all the nodes in \( Z(\tau) \) according to
their numbers of patterns.
IV. Experimental Results

4.1 Experiment Settings

Although any clustering algorithm for relational data can be used to obtain the final clustering from the co-association matrix, we focus our experiments on the framework of evidence-accumulation clustering (EAC) as described in [12]. The common k-means algorithm is used to generate the base clusterings, with the number of clusters $k$ in each clustering randomly selected from an interval $[k_{\text{min}}, k_{\text{max}}]$. The actual values of $k_{\text{min}}$ and $k_{\text{max}}$ are dataset dependent. However, we use a large range to ensure that the results are not too sensitive to the choice of an optimal $k$. The clustering runs are initialized using a randomly selected subset of the data points as the initial prototypes. The experimental results presented here, unless otherwise noted, use EAC-AL (EAC plus hierarchical agglomeration with average linkage for obtaining the final clustering), which is the best overall performer in [12]. The final cluster label of a pattern $x$ is just the final cluster label assigned to the node $z$ where $X(z)$ contains $x$.

The quality of the final clustering is evaluated by matching the final cluster labels with the ground-truth cluster labels of the patterns. The ground truth consists of the class labels of real data sets and the cluster labels used when generating the synthetic data sets. We use the Hungarian algorithm to find the optimal assignment (the one that results in the largest number of correctly labeled patterns) between the two sets of labels. We then use the ratio of correctly labeled patterns using this optimal assignment as the clustering accuracy measure. We use $Q(\tau)$ to represent the clustering accuracy at $\tau$.

For the experiments described below, the results are always averaged over 20 ensembles. Unless noted otherwise, the user-specified parameters are $H=20$, $n_{\text{des}}=32$, and $\gamma=1$ (without node
reduction) or 0.9 (with node reduction).

4.2. Data Sets

Table II gives a summary of the data sets used in our experiments, including the intervals \([k_{\text{min}}, k_{\text{max}}]\) used. Here \(N\) is the number of patterns, \(L\) is the data dimensionality (number of features) and \(k^*\) is the "natural" (ground-truth) number of clusters, taken as the number of classes for the real data sets and as the number of clusters used for generating a synthetic data set. The following provides more detailed description to these data sets:

- **Spherical5**: Five touching spherical clusters with 50 patterns each in a 16-d space. The 5 clusters have the same size and their centroids all fall on the plane of the first two dimensions.
- **Half-rings**: Two half rings with 200 and 600 patterns, respectively. The distribution of patterns is similar to the half-rings data set in [12].
- **3-Rings**: Three concentric circles with 100, 400 and 400 patterns, respectively. The distribution of patterns is similar to the 3-rings data set in [12].
- **8d5k** [10]: This is a synthetic data set of 5 ellipsoidal clusters with 200 patterns each in an 8-d space. The 5 clusters are well separated and have identical covariance matrix.
- **Opt-digits** (Optical Recognition of Handwritten Digits): A 10-class (each for one digit), 3823-pattern, 64-d data set from the UCI Machine Learning Repository [34].
- **Pen-digits** (Pen-Based Recognition of Handwritten Digits): A 10-class (each for one digit), 10992-pattern, 16-d data set from the UCI Machine Learning Repository [34].

The first 3 synthetic data sets (projected onto the first two dimensions if \(L>2\)) are shown in Fig. 4. The last two (Opt-digits and Pen-digits) are selected because they are large for co-association matrices. For example, authors of [12] only use the first 100 patterns of each of the 10 classes in Opt-digits.
4.3. Experiments on the Properties of CA-Trees

We start by showing in Fig. 5 how $N_Z(\tau)/N$ varies with $\tau$ for several data sets. The value of $\tau$ ranges from 0 to $H-1$. When $\tau$ is zero, each node in $Z(\tau)$ is a core group. Therefore, $N_Z(\tau=0)$ is the total number of core groups. We can see that $N_Z(\tau)/N$ is down to 0.1 or much lower at, say, $\tau=H/2$, resulting in significant computational saving for the processing of the co-association matrix.

In Fig. 5 we actually have two curves in each plot corresponding to $n_{des}=32$ (blue curve) and $n_{des}=\infty$ (magenta curve), respectively. The latter case corresponds to using the exact node sizes computed according to (5) to (7). In each plot, either the two curves are identical or they are only slightly different for larger $\tau$. The small difference indicates that using $n_{des}=32$ results in a good estimation of node sizes for our experimental settings. However, we want to note that a larger

---

Table II. Summary of Data Sets

<table>
<thead>
<tr>
<th>Data Set</th>
<th>N</th>
<th>L</th>
<th>$k^*$</th>
<th>$[k_{min}, k_{max}]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spherical5</td>
<td>250</td>
<td>16</td>
<td>5</td>
<td>[4, 10]</td>
</tr>
<tr>
<td>Half-rings</td>
<td>800</td>
<td>2</td>
<td>2</td>
<td>[8, 20]</td>
</tr>
<tr>
<td>3Rings</td>
<td>900</td>
<td>2</td>
<td>3</td>
<td>[20, 50]</td>
</tr>
<tr>
<td>8d5k [10]</td>
<td>1000</td>
<td>8</td>
<td>5</td>
<td>[4, 10]</td>
</tr>
<tr>
<td>Opt-digits</td>
<td>3823</td>
<td>64</td>
<td>10</td>
<td>[10, 40]</td>
</tr>
<tr>
<td>Pen-digits</td>
<td>10992</td>
<td>16</td>
<td>10</td>
<td>[10, 40]</td>
</tr>
</tbody>
</table>

---

Fig. 4. Three synthetic data sets used in our experiments: (a) Spherical5; (b) Half-rings; (c) 3rings.
$n_{des}$ may be needed if significantly more base clusterings are used, leading to a deeper tree. This is because the estimation error accumulates through the recursive processing of the tree, which is the reason why the difference is more evident at larger $\tau$.

Fig. 5 does not tell us how to select an appropriate threshold. For this purpose, we are more interested in how the clustering accuracy varies with $\tau$. Fig. 6 contains the plots of $Q(\tau)$ versus $\tau$ for the same data sets as in Fig. 5. Here we focus on the case where we select the final clustering with the known number ($k^*$) of clusters during hierarchical agglomeration. To help us choose an appropriate threshold, in each plot we also show a horizontal dashed line indicating the clustering accuracy at $0.9Q(0)$, meaning a 10% degradation from the accuracy obtained by directly using the core groups. For example, a common threshold of $\tau=10$ ($\tau H=0.5$) is applicable for all 6 data sets if 10% is considered the acceptable level of degradation. To err on the conservative side, $0.4H$ and $0.2H$ seem to be reasonable choices for $\gamma=1$ and $\gamma=0.9$.
respectively. We still get most of the benefit as the most significant drop of $N(\tau)$ occurs at small values of $\tau$, which is evident in Fig. 5.

A closer examination of Fig. 6 seems to indicate that there are two types of behaviors in terms of how clustering accuracy varies with $\tau$. For 4 of the 6 data sets (Fig. 6(a), (d), (e), and (f)), the curves are quite flat and $Q(\tau)$ only drops slightly (close to or less than 10%) even at $\tau = H-1$. On the other hand, Fig. 6(b) and (c) seem to exhibit some critical threshold, after which $Q(\tau)$ drops dramatically. This can be explained by the fact that these two data sets have elongated curvilinear clusters, and $N_\ell(\tau)$ at this "critical threshold" approximately correspond to the number of representative points needed to preserve the cluster information in the data. $N_\ell(\tau)$ at $\tau =10$ is 21 and 55 for data sets half-rings and 3-rings, respectively. This observation seems to suggest two different strategies for threshold selection if we have some information about the type of clusters: larger thresholds to get smaller $N_\ell(\tau)$ and hence less computation for compact and
spherical clusters, and smaller thresholds to ensure acceptable accuracy for clusters that are curvilinear or otherwise have non-compact shapes. While the knowledge about cluster shapes may not be available, the process of node reduction can provide us with some useful information on this. This is further explained later in subsection 4.6.

Our main goal is the reduction of complexity by using only the selected nodes to build the co-association matrix. It is therefore important to analyze how \( N_Z(\tau) \) scales with \( N \). For this purpose, we generate additional data sets that have the same distributions as several synthetic data sets but different \( N \). For the three data sets from external sources (8d5k, Opt-digits and Pen-digits), the additional data sets are obtained by random subsampling. Fig. 7 displays the logarithmic plots of \( N_Z(\tau) \) versus \( N \) without node reduction at a few different \( \tau \) for the Spherical5 and Pen-digits data sets. The nearly linear plots suggest that it is reasonable to assume that

\[
N_Z(\tau) \propto N^{\alpha(\tau)},
\]

where \( \alpha(\tau) \) is data-set dependent. The complexity associated with the co-association matrix becomes \( O(N^{2\alpha(\tau)}) \) instead of \( O(N^2) \). As long as \( \alpha(\tau) < 1 \), we have lower than quadratic complexity and the cluster ensemble algorithm using our selected nodes will scale better to large data sets than the original version that uses all the patterns in the data set.

In Tables III and IV we list more \( \alpha(\tau) \) values for additional data sets using \( \gamma = 1 \) and \( \gamma = 0.9 \), respectively. It is evident that \( \alpha(\tau) \) decreases fairly fast as the threshold increases. This is an indication that \( N_Z(\tau) \) is more affected by \( N \) at very low thresholds and by the actual distribution at higher thresholds. At our recommended thresholds (0.4\( H \) for \( \gamma = 1 \) and 0.2\( H \) for \( \gamma = 0.9 \)), all the \( \alpha(\tau) \) values are less than 0.5, resulting in the complexity associated with the co-association matrix being below \( O(N) \). A comparison between these two tables also shows that node reduction leads to substantially lower complexity at similar thresholds. Even when comparing \( \tau = 0.4H \) for
\(\gamma = 1\) and \(\tau = 0.2H\) for \(\gamma = 0.9\), four of the \(\alpha(\tau)\) values for \(\gamma = 0.9\) are much less than the corresponding values for \(\gamma = 1\), and for the other two (3Rings and 8d5k) the \(\alpha(\tau)\) values are already close to zero. This is an indication that node reduction is an attractive option when we need to process very large data sets. Lower \(\gamma\) is expected to lead to even lower \(\alpha(\tau)\) but can also result in lower clustering accuracy because fewer nodes are used to represent the data distribution.

Table III. Values of \(\alpha(\tau)\) at Different Thresholds (\(\gamma = 1\))

<table>
<thead>
<tr>
<th>Data Set</th>
<th>(N_{\text{min}})</th>
<th>(N_{\text{max}})</th>
<th>(\tau = 0)</th>
<th>(\tau = 0.1H)</th>
<th>(\tau = 0.2H)</th>
<th>(\tau = 0.3H)</th>
<th>(\tau = 0.4H)</th>
<th>(\tau = 0.5H)</th>
<th>(\tau = 0.6H)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spherical5</td>
<td>250</td>
<td>4000</td>
<td>0.73</td>
<td>0.64</td>
<td>0.55</td>
<td>0.46</td>
<td>0.42</td>
<td>0.38</td>
<td>0.32</td>
</tr>
<tr>
<td>Half-rings</td>
<td>400</td>
<td>6400</td>
<td>0.40</td>
<td>0.23</td>
<td>0.18</td>
<td>0.17</td>
<td>0.16</td>
<td>0.16</td>
<td>0.12</td>
</tr>
<tr>
<td>3Rings</td>
<td>450</td>
<td>7200</td>
<td>0.40</td>
<td>0.18</td>
<td>0.10</td>
<td>0.07</td>
<td>0.05</td>
<td>0.03</td>
<td>0.02</td>
</tr>
<tr>
<td>8d5k</td>
<td>63</td>
<td>1000</td>
<td>0.63</td>
<td>0.42</td>
<td>0.18</td>
<td>0.08</td>
<td>0.03</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>Opt-digits</td>
<td>239</td>
<td>3823</td>
<td>0.63</td>
<td>0.50</td>
<td>0.42</td>
<td>0.38</td>
<td>0.35</td>
<td>0.32</td>
<td>0.28</td>
</tr>
<tr>
<td>Pen-digits</td>
<td>687</td>
<td>10992</td>
<td>0.57</td>
<td>0.52</td>
<td>0.47</td>
<td>0.43</td>
<td>0.40</td>
<td>0.37</td>
<td>0.34</td>
</tr>
</tbody>
</table>

Fig. 7. Logarithmic plots of the number of selected nodes versus \(N\) at 4 different thresholds. Data sets: (a) Spherical5, (b) Pen-digits.
Table IV. Values of $\alpha(\tau)$ at Different Thresholds ($\gamma=0.9$)

<table>
<thead>
<tr>
<th>Data Set</th>
<th>$N_{min}$</th>
<th>$N_{max}$</th>
<th>$\tau=0$</th>
<th>$\tau=0.1H$</th>
<th>$\tau=0.2H$</th>
<th>$\tau=0.3H$</th>
<th>$\tau=0.4H$</th>
<th>$\tau=0.5H$</th>
<th>$\tau=0.6H$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spherical5</td>
<td>250</td>
<td>4000</td>
<td>0.65</td>
<td>0.42</td>
<td>0.25</td>
<td>0.20</td>
<td>0.17</td>
<td>0.16</td>
<td>0.12</td>
</tr>
<tr>
<td>Half-rings</td>
<td>400</td>
<td>6400</td>
<td>0.22</td>
<td>0.12</td>
<td>0.09</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
<td>0.04</td>
</tr>
<tr>
<td>3Rings</td>
<td>450</td>
<td>7200</td>
<td>0.26</td>
<td>0.13</td>
<td>0.07</td>
<td>0.05</td>
<td>0.03</td>
<td>0.02</td>
<td>0.02</td>
</tr>
<tr>
<td>8d5k</td>
<td>63</td>
<td>1000</td>
<td>0.51</td>
<td>0.28</td>
<td>0.13</td>
<td>0.08</td>
<td>0.04</td>
<td>0.02</td>
<td>0.02</td>
</tr>
<tr>
<td>Opt-digits</td>
<td>239</td>
<td>3823</td>
<td>0.54</td>
<td>0.26</td>
<td>0.07</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>Pen-digits</td>
<td>687</td>
<td>10992</td>
<td>0.28</td>
<td>0.14</td>
<td>0.10</td>
<td>0.09</td>
<td>0.09</td>
<td>0.09</td>
<td>0.08</td>
</tr>
</tbody>
</table>

4.4. Experiments with Very Large Data Sets

To further demonstrate the ability of CA-tree to make co-association based cluster ensembles scalable to very large data sets, we include here the example of clustering the pixels in an image. Fig. 8(a) is a 640x480 image with a total of over $3\times10^5$ pixels. Using the RGB values of each pixel as a pattern to be clustered, we have a data set almost 30 times larger than the largest data set (Pen-digits) used in our experiments so far. We apply the EAC-AL algorithm with CA-tree using the following parameters: $H=10$, $k_{min}=3$, $k_{max}=6$, $\tau=0.2H$, $\gamma=0.9$, and $k^*=3$ for the three main regions (sky, road, and greenery). The result is displayed in Fig. 8(b) where each color represents a cluster. We can see that the clustering result is quite accurate except for where the treetops meet the sky. Using 1000 manually labeled pixels randomly selected from the image, we estimate the clustering accuracy to be 0.979. This experiment clearly indicates the usefulness of CA-tree for applying cluster ensembles to very large data sets.

4.5. Comparisons with the Prototype-Based Method

We are interested in comparing our method with the prototype-based method in terms of how well the information of the original data set is condensed into the set of selected nodes (our method) or pre-clusters (prototype-based method). For the prototype-based method, three pre-clustering approaches are implemented for comparison: (1) random sampling; randomly
selecting a subset of data points as the prototypes, (2) k-means, (3) k-medoids. Both k-means and k-medoids are randomly initialized for each ensemble.

We base the comparison on how $Q$, the clustering accuracy, varies with $N_Z$. For the prototype-based methods, $N_Z$ is the number of prototypes, and we obtain the results of these methods at several pre-specified $N_Z$. For the CA-tree, we obtain results at different $N_Z$ by varying $\tau$. Both similar clustering accuracy at smaller $N_Z$ and higher accuracy at similar $N_Z$ indicate a better ability for information condensation. The plots are displayed in Fig. 9. Each plot contains five curves: two for CA-tree with $\gamma=1$ and $\gamma=0.9$, respectively, and three for the prototype-based methods. Our method with $\gamma=0.9$ clearly outperforms the prototype-based methods for all the data sets except for in Fig. 9(e) where they are comparable.

Fig. 10 is similar to Fig. 9 except that the final clustering is selected according to the maximum-lifetime criterion in hierarchical agglomeration. Again our method with $\gamma=0.9$ is the best performer except for in Fig. 10(b) and (c), where the clustering accuracy is extremely low for all the methods.
4.6. Node Reduction and Cluster Shapes

When comparing the results with and without node reduction in Fig. 9 and 10, the effect of node reduction seems to range from significantly better (Opt-digits and Pen-digits), slightly better (Spherical5 and 8d5k), to somewhat worse (Half-rings and 3rings). To better understand
these differences, we plot in Fig. 11 the ratios $N_Z(\tau)/N_Z^*(\tau)$. (As mentioned previously, when we need to disambiguate between the two, $N_Z(\tau)$ and $N_Z^*(\tau)$ represent the values with and without node reduction, respectively.) In general, the smaller the ratio, the stronger the information condensation is as more nodes are excluded. The minimal ratio for a particular data set seems to be a useful indicator of the effect of node reduction. These minimal ratios are 0.189 (Pen-digits), 0.267 (Opt-digits), 0.433 (Spherical5), 0.506 (8d5k), 0.557 (Half-rings), and 0.671 (3rings). We can see that smaller minimal ratios somewhat correlate with the improvements from node reduction, although only qualitatively. As a starting point, we believe that this minimal ratio can be a useful rough guideline regarding whether to use node reduction. For example, node reduction is used when the minimal ratio is below a threshold of 0.5.

To better understand the information in the ratio $N_Z(\tau)/N_Z^*(\tau)$, let us consider the following
scenarios: When we have compact actual cluster structure in the data, the regions between clusters have low data density. The "intersections" between base clusters in these regions therefore are likely to have few data points, forming small groups in the CA-tree. Such are the nodes that are merged into other nodes in node reduction. Since these groups are small, we can merge many nodes at a given $\gamma$, giving a smaller ratio $N_Z(\tau)/N_Z^*(\tau)$. On the other hand, for elongated actual clusters, the intersections between (compact) base clusters have similar density as the regions inside the base clusters, resulting into more groups with similar data counts. Node reduction in this case gives larger $N_Z(\tau)/N_Z^*(\tau)$ (i.e., fewer merged nodes) at a given $\gamma$. Overall, these arguments can help to explain our observation that the two datasets with known elongated cluster structures (Half-rings and 3rings) have the largest minimum $N_Z(\tau)/N_Z^*(\tau)$ ratios, and support our idea that this ratio contains information about actual cluster shapes. We also suspect that there is more useful information regarding the underlying cluster shapes that can be deduced from the statistics of pattern counts in the nodes at different thresholds, a point that requires additional investigation.

V. Computational Complexity

5.1. Analysis of Complexity

In the following we summarize the computational complexity involved in the various steps of our proposed method:

1) Tree construction: $O(N \cdot H)$;

2) Node size and representative determination: $O(N \cdot H \cdot (k_{\text{max}} + n_{\text{des}})^2)$;

3) Computation of the co-association matrix: $O(H \cdot N_Z(\tau)^3)$;

4) If node reduction is used, the selection of the retained nodes: $O(N_Z^*(\tau) \cdot \log N_Z^*(\tau))$;
5) If node reduction is used, the final cluster label assignment for the excluded nodes:

\[ O(N_Z^* \cdot \tau \cdot H \cdot k_{h(max)}) \]

Here we do not include the complexity involved in generating the base clusterings and extracting the final clustering, as there are many different algorithms applicable to these two steps and their differences are not our focus.

The combined expression for the complexity is

\[ O(N \cdot H \cdot (k_{max} + n_{des})^2 + N^{2 \alpha(\tau)} \cdot H + N^{\alpha^*(\tau)} \cdot (\log N + H \cdot k_{max})) \]  

after substituting (17) for \( N_Z(\tau) \) and an analogous expression

\[ N_Z^*(\tau) \propto N^{\alpha^*(\tau)}, \]

for \( N_Z^*(\tau) \). We have also used the user-specified parameter \( k_{max} \) as the upper bound for \( k_{h(max)} \).

Since we expect that \( \alpha^*(\tau) < 1 \), the third term in (18) is always less than the first term and can be dropped. We end up with the expression

\[ O(N \cdot H \cdot (k_{max} + n_{des})^2 + N^{2 \alpha(\tau)} \cdot H). \]

If we have \( 2\alpha(\tau) \leq 1 \), which is the case in our experiments except for very low thresholds (see Table IV), then the overall complexity is linear with respect to \( N \). Even when this condition is not satisfied, the savings over the original quadratic complexity is still very significant for large \( N \). In addition, this lower complexity is even more important when the algorithm for final clustering extraction has higher than quadratic complexity. Such examples include average or complete linkage, both having \( O(N^2 \log N) \) complexity using existing efficient hierarchical agglomeration algorithms [35][36].

Regarding space complexity, the co-association matrix now has \( N^{2 \alpha(\tau)} \) elements. The tree itself has at most \( 2N \) nodes. Similar to time complexity, the space complexity is linear with respect to \( N \) when \( 2\alpha(\tau) \leq 1 \), and is \( O(N^{2 \alpha(\tau)}) \) otherwise.
5.2. Experiments on Computation Time

This subsection consists of experimental results regarding the actual computation time spent in constructing a cluster ensemble. The purpose is to provide empirical evidences to the theoretical claim of reduced computational complexity in the previous subsection. The experimental environment is Matlab R14 running on a 3.0GHz Pentium IV PC with 1GB of memory, while the majority of time-consuming work is implemented in C. The timing data is obtained with the Matlab profiler.

Specifically, we compare the required execution time when the data structure for the ensemble (such as the co-association matrix) is built using one of the following (referred to as methods (A)-(D) in this subsection):

(A) the original data set;

(B) the core groups, corresponding to CA-tree with $\tau = 0$;

(C) the groups selected from the CA-tree with $\tau = 0.2$;

(D) the same as (C) plus node reduction ($\gamma = 0.9$).

The experimental settings are the same as those in subsection 4.3 except that we use $H=10$. Two synthetic data sets, Spherical5 and 3rings, are used here. However, to emphasize the performance comparison for larger data sets, we increase the numbers of samples in these two data sets to 2000 and 1800, respectively, while maintaining the same distributions as the original data sets.

Table V list the execution time (in seconds) and clustering accuracy averaged over 20 ensembles. Here $t_{total}$ represents the total execution time, including the time spent on obtaining the base clusterings. On the other hand, $t_{CE}$ includes only the time for extracting the final clustering from available base cluster labels. While our focus is on the improvement of $t_{CE}$, we
list \( t_{total} \) as well to provide the information about the total time requirement when the base clustering is not available.

In addition to EAC-AL, we also list the results for three well-known graph-based cluster ensemble algorithms: HBGF [26], CSPA [21] and MCLA [21]. We use METIS [37] as the method that partitions the graph into final clusters. Among the three, only CSPA is based on co-association matrices. The computational complexities of HBGF and MCLA are both linear with respect to \( N \). The motivation here is to see whether and how much performance improvement is achievable by using CA-tree with such methods.

The improvement in execution time is most significant with co-association based methods (EAC-AL and CSPA) as expected. For HBGF and MCLA, \( t_{CE} \) is also reduced by about 60-75 percent from method (A) to method (D), with little loss of clustering accuracy. This indicates that CA-tree is still useful in improving the efficiency and scalability of cluster ensemble algorithms that are not based on co-association matrices. The time used to obtain the base clusterings (approximately \( t_{total} - t_{CE} \)) stays approximately constant for all the methods, which is not surprising. For methods (C) and (D), we have reduced \( t_{CE} \) so much that the time for base clusterings mostly dominate \( t_{total} \). We can also see that the improvement in execution time obtained with node reduction (from (C) to (D)) is not as significant that the improvement obtained with the CA-tree itself (from (A) to (C)). This is because the time spent on processing node reduction is not negligible compared with clustering the already small set of groups from method (C). However, the improvement from node reduction is more significant for larger datasets. For example, if Spherical5 is scaled to \( N=10000 \), \( t_{CE} \) for EAC-AL becomes 0.61 and 0.09 for methods (C) and (D), respectively, a factor of about 6.7, compared to about 1.9 (0.11/0.06) when \( N=2000 \).
One more important observation can be made about the results for 3rings (and more generally, datasets with non-compact clusters). We see that the clustering accuracies obtained with the three graph-based algorithms range from 0.74 to 0.78 and are far below the near perfect accuracy achieved by EAC-AL. This is previously known in [12] and is a major motivation behind using the inefficient hierarchical agglomeration algorithms to get the final clusterings. However, when considering methods (C) and (D), EAC-AL outperforms the three graph-based algorithms in both execution time and accuracy. In a word, CA-tree significantly alleviates the problem of computational cost for algorithms such as EAC-AL, allowing us to use the most appropriate cluster ensemble algorithms on different datasets without much worry about efficiency.

Table V. Comparison of Execution Time and Clustering Accuracy

<table>
<thead>
<tr>
<th>Data set</th>
<th>Spherical5 (N = 2000)</th>
<th>3rings (N = 1800)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>$t_{total}$</td>
<td>$t_{CE}$</td>
</tr>
<tr>
<td>EAC-AL(A)</td>
<td>111.54</td>
<td>111.29</td>
</tr>
<tr>
<td>EAC-AL(B)</td>
<td>1.19</td>
<td>0.94</td>
</tr>
<tr>
<td>EAC-AL(C)</td>
<td>0.36</td>
<td>0.11</td>
</tr>
<tr>
<td>EAC-AL(D)</td>
<td>0.31</td>
<td>0.06</td>
</tr>
<tr>
<td>CSPA(A)</td>
<td>7.76</td>
<td>7.51</td>
</tr>
<tr>
<td>CSPA(B)</td>
<td>0.68</td>
<td>0.43</td>
</tr>
<tr>
<td>CSPA(C)</td>
<td>0.40</td>
<td>0.16</td>
</tr>
<tr>
<td>CSPA(D)</td>
<td>0.37</td>
<td>0.12</td>
</tr>
<tr>
<td>HBGF(A)</td>
<td>0.51</td>
<td>0.26</td>
</tr>
<tr>
<td>HBGF(B)</td>
<td>0.43</td>
<td>0.18</td>
</tr>
<tr>
<td>HBGF(C)</td>
<td>0.38</td>
<td>0.14</td>
</tr>
<tr>
<td>HBGF(D)</td>
<td>0.38</td>
<td>0.13</td>
</tr>
<tr>
<td>MCLA(A)</td>
<td>0.46</td>
<td>0.22</td>
</tr>
<tr>
<td>MCLA(B)</td>
<td>0.41</td>
<td>0.16</td>
</tr>
<tr>
<td>MCLA(C)</td>
<td>0.39</td>
<td>0.14</td>
</tr>
<tr>
<td>MCLA(D)</td>
<td>0.39</td>
<td>0.15</td>
</tr>
</tbody>
</table>

Note: (A), (B), (C) and (D) represent different settings for using the CA-tree in building the co-association matrix; please refer to the text for more detail.
VI. Conclusions

In summary, in this paper we propose the CA-tree, a new data structure for improving the scalability of ensemble clustering algorithms for large data sets, especially those based on co-association matrices. When cut at a particular threshold for node sizes, a set of nodes are selected to approximate the data set in the space of base cluster labels. Subsequent ensemble clustering algorithms are applied to this set of selected nodes instead of the original patterns, leading to reduced computational complexity. A node reduction scheme is also proposed to further reduce the amount of computation by excluding nodes that contain few patterns. As demonstrated by our experiments with various data sets, we can often achieve complexity of lower than linear to $N$ for co-association matrix processing with little loss in clustering accuracy, and the complexity for building the tree is linear to $N$. This improvement of efficiency is dramatic compared to the $O(N^2)$ or higher complexity of popular co-association based algorithms such as EAC and CSPA, therefore significantly extending their applicability to more clustering problems.

The CA-tree has several advantages over the prototype-based method that was previously employed to help solve the scalability problem. In addition to overall better clustering accuracy, our method does not require the guesswork involved in determining the number of prototypes to use. Because the only information required for building the CA-tree is the base cluster labels, it is applicable to distributed multi-view clustering when a data set does not have all the features, and is able to incorporate existing base clusterings. These are two well-known ensemble clustering scenarios that the prototype-based method is unable to address.

One important element of our method, the power-law relation between $N$ and $N_Z(\tau)$ (see (17)), is currently supported by empirical results. In the future, we intend to provide a deeper
understanding of such a relation from theoretical perspectives, as well as to investigate factors that affect $\alpha(\tau)$. Also of theoretical interest is the information regarding the properties of the actual clusters that may exist in the statistics of pattern counts in the nodes at various thresholds, such as to tell whether the clusters are elongated or compact. While current results seem to hint at the existence of such information (see subsection 4.6), more study, both experimental and theoretical, will be needed to provide a clearer picture. Another possible subject for future study is the extension of our method to fuzzy cluster ensembles, a subject that has attracted increasing interests in recent years.

References


