Tree reduced ensemble clustering and distances between cluster trees based on a graph algebraic framework

R.W. Oldford
University of Waterloo
Waterloo, Ontario, Canada

W. Zhou
University of Waterloo
Waterloo, Ontario, Canada

February 20, 2014

Abstract

The problem of finding multiple, possibly overlapping clusters is shown to share mathematical structure with the problem of combining the outcomes from multiple clusterings. Both problems can be cast in terms of sets of graphs, which we call graph families. A graph algebraic framework is developed through illustrative examples and formally presented. By way of this framework, a new multi-clustering method we call tree reduced ensemble clustering is developed that is applicable to the outcomes of any combination of clustering methods. Several examples including $k-$means, model-based clustering, single-linkage and complete-linkage clustering are used to illustrate and to develop the graph algebraic framework for clustering and multi-clustering. A new distance between cluster trees is presented and shown to be useful in the comparative study of clustering outcomes.

Keywords: cluster trees, graph families, ensemble clustering, multi-clustering, partitions, hierarchical clustering, graph-based clustering, consensus clustering, distance between cluster trees

1 Introduction

In this paper we develop a new ensemble method for combining multiple clustering outcomes. The method is agnostic about the source of the clusterings and so is not tied to any clustering method such as k-means. The ensemble method, TREC, or *tree reduced ensemble clustering*, produces a single cluster tree as its output. The result is not a dendrogram as would be produced by a typical hierarchical clustering method like single linkage, but rather a *cluster tree* similar in structure to the density cluster trees described in (Hartigan 1985) though requiring no such density interpretation.

To develop the methodology, we cast the multiple clustering problem as one of summarizing the cluster structure provided by a set of graphs on the same vertex set. We call a set of such graphs a *graph family* and introduce some formal mathematical structure that describes the operations performed on a graph family to produce a cluster tree.

The methodology is developed via simple examples and then applied on data where clustering outcomes can be from partition methods like k-means and Gaussian mixture model based clustering, hierarchical methods like single and complete linkage, or any mixture of partition and hierarchical methods. The outcomes need not even be generated by any formal method or algorithm whatsoever. Two data sets, one generated from a mixture of three Gaussians and the other an artificial pair of spirals are clustered by the various methods to motivate and to illustrate the methodology.

The methodology used to produce the ensemble method is then formalized mathematically and the nature of the various operations briefly described. More importantly, these operations, their mathematical characteristics, and the sets on which they operate together form a graph algebraic framework in which the problem of multiple clustering may be cast. Various sets of graph families can be placed within the framework and connected to isomorphic spaces of matrices and of cluster trees. The TREC methodology is seen to provide a means of projecting the outcome (or outcomes) of any clustering method into a space of

cluster trees. Moreover, a metric is defined on a space of matrices which is isomorphic to the space of cluster trees. This induces a corresponding metric on the cluster trees which can be used to compare the the cluster trees and hence the characteristics of the clustering methods which led to them.

The paper separates into two major parts. The first is Section 2 where the proposed methodology is developed via illustrative examples. Relevant literature review is used to motivate the approach. Once developed, the methodology is applied to the example data sets. These examples show how the TREC methodology works. The second major part is Section 3 where the formal graph algebraic framework is mathematically abstracted and summarized. All necessary proofs can be found in the Appendix. In Section 3 the distance measure is also introduced and applied to the cluster trees constructed in Section 2.4 for the Gaussian mixture data. These distances are used in multidimensional scaling to position the various cluster trees in a two dimensional space. This provides a quick visual means to assess various characteristics of the resulting cluster trees (TREC performs well).

The paper ends with some brief concluding remarks and future directions in Section 4.

2 Developing the methodology

In this section we develop our methodology through the exploration of several examples.

Section 2.1 considers the problem of multiple clusterings and how others have proposed to address it. This review suggests that a weighted adjacency matrix, A_ω, plays an essential role in these solutions and hence that a graph-theoretic approach could be a helpful underpinning to any such investigation.

In Section 2.2, we examine the value of using this matrix by applying it to multiple clustering of a generated data set where the underlying true cluster structure is known (a Gaussian mixture and using outcomes from different k-means clustering outcomes). This examination, however, indicates that the raw A_ω has shortcomings as a cluster summary.

To overcome these, in Section 2.3 the problem of multiple clustering is cast in terms of families of graphs (each graph representing a clustering). The goal becomes to transform the graph family so that its resulting weighted adjacency matrix, A_ρ, is a more useful summary of the multiple clusterings. Working through simple examples, Section 2.3 develops and illustrates the sequence of transformations. This sequence defines the ensemble method we call *tree reduced ensemble clustering*, or *TREC*.

In Section 2.4 TREC is illustrated by applying the method to several examples. In addition to the several outputs from k-means considered earlier, in this section we also combine the output from several other clustering methods including Gaussian mixture model clustering, and simple and complete linkage. The method applies universally to the outcome of all methods which produce one or more clusterings.

2.1 Motivation – multiple clusterings

The problem of multiple clusterings arises for many reasons. All hierarchical clustering methods inherently produce multiple clusterings as a set of nested partitions (e.g. see Hartigan 1975). Other methods intentionally produce multiple non-nested clusterings to capture different facets of similarity between objects (a notable early example being `ADCLUS`, from the psychometric literature – Shepard and Arabie 1975, 1979). Different methods have often arisen from differing philosophies or views as to the purpose of the clustering. These include, for example, teasing apart different notions of humanly perceived similarity between stimuli (e.g. Tversky 1977; Shepard and Arabie 1979), fitting mixture distributions to sample measurements (e.g. Fraley and Raftery 1999), optimizing an objective function designed to capture "natural" spatial structure in dimensional data (e.g. k-means, MacQueen 1967), determining high density modes from a sample (e.g. Hartigan 1981; Ester et al. 1996; Stuetzle 2003), and determining approximate graph components in a similarity graph (e.g. Donath and Hoffman 1973; Meila and Shi 2001), to mention the more common approaches. Multiple clusterings can also arise from multiple local optima (e.g. k-means, MacQueen 1967) or simply from multiple values of some "tuning parameter" (e.g. again k-means, Ashlock et al. 2005). Sometimes multiple clusterings are even induced by resampling (e.g. Topchy et al. 2004) so that they can be later combined. Finally, in practice one might routinely choose a variety of methods, knowing in advance that each was

sensitive to a different kind of cluster structure (e.g. k-means and complete linkage prefer globular clusters while single-linkage will favour stringy clusters), so as to cover a wide range of possibilities.

Multiple clusterings might be summarized by a single partition (e.g. as in Strehl and Ghosh 2002; Dimitriadou et al. 2001), by a single tree (as does every hierarchical clustering or more recently as does Fred and Jain 2002), or even by several trees as in Carroll and Corter (1995). The multiple clusterings themselves could be used to produce a new similarity matrix which is then itself used as input to some clustering algorithm, yielding whatever is the natural output of that method and begging the question as to which algorithm it would be best to use (e.g. Fred and Jain 2002; Ashlock et al. 2005; Strehl and Ghosh 2002) – curiously, one clustering algorithm might be recommended to produce multiple clusterings and a different one for their combination (e.g. Fred 2001; Fred and Jain 2002; Ashlock et al. 2005). Alternatively, the cluster ensemble problem could be cast as another optimization problem whereby a final partition would be chosen to maximize some criterion, like a normalized mutual information measure (e.g. Strehl and Ghosh 2002), involving the given collection of clusterings. Either way, a multiple clustering is replaced with a single clustering problem which has access to the similarity information accumulated across the original clusterings.

In what follows, we take a somewhat different approach. We begin with a set $\mathcal{G}$ of m graphs G_k and their corresponding adjacency matrices A_k, for $k = 1, 2, \ldots, m$. We might reasonably summarize this set by the sum of these adjacency matrices $A_{\mathcal{G}} = \sum_{k=1}^{m} A_k$. Perhaps not surprisingly, this simple sum is related to a popular means of summarizing cluster ensembles.

To see this, first consider how `ADCLUS` (Shepard and Arabie 1975, 1979) constructs multiple, possibly overlapping clusters. The approach is based on approximating an original similarity matrix, $S^* = [s^*_{ij}]$, with a new one, $S = [s_{ij}]$, constrained to have the following structure:

$$s_{ij} = \sum_{k=1}^{K} w_k p_{ik} p_{jk}.$$

Here, K is the total number of (possibly overlapping) clusters, w_k weights the importance (or "psychological salience" as in Shepard and Arabie 1979) of cluster k and p_{ik} is 1 if i appears in cluster k and 0 otherwise. For n objects, the kth cluster corresponds to a binary vector $\mathbf{p}_k = (p_{1k}, p_{2k}, \ldots, p_{nk})^T$. The `ADCLUS` derived $n \times n$ similarity matrix $S = [s_{ij}]$ can also be written as

$$S = \sum_{k=1}^{K} w_k \mathbf{p}_k \mathbf{p}_k^T = PWP^T \tag{1}$$

where now W is a $K \times K$ diagonal matrix of weights and P is the $n \times K$ matrix whose kth column is $\mathbf{p}_k$. Note that, although there is no restriction on the diagonal terms of S in this representation, only the off diagonal similarities are of interest in clustering.

Multiple clusterings can use the same representation. For example, suppose we have r partitions of n objects. Following Strehl and Ghosh (2002), the kth partition is represented as a binary membership indicator matrix, $P^{(k)}$ having n rows and as many columns as there are clusters in that partition. Each column of $P^{(k)}$ is a binary vector representing a single cluster with 1 indicating membership in that cluster. A similarity matrix is constructed as

$$S = \sum_{k=1}^{r} P^{(k)} (P^{(k)})^T = PP^T \tag{2}$$

where $P = [P^{(1)}, \cdots, P^{(r)}]$. This sum is equivalent to a voting mechanism where each cluster contributes one vote to every pair of objects within it. Subsequent clustering from this similarity matrix is the basis of the clustering ensemble methods used in Strehl and Ghosh (2002), Fred and Jain (2002), and Ashlock et al. (2005). Again, only the off-diagonal terms are of value.

Each of the above equations can be re-expressed in terms of adjacency matrices A_k. For equation (1), we note that $\mathbf{p}_k \mathbf{p}_k^T$ is a symmetric $n \times n$ matrix whose off-diagonal elements correspond to the adjacency matrix,

A_k, of the graph on n nodes with edges only between all pairs in the kth cluster. Equation (1) becomes

$$S = \sum_{k=1}^{K} w_k \mathbf{p}_k \mathbf{p}_k^T = \sum_{k=1}^{K} w_k J_k + \sum_{k=1}^{K} w_k A_k \tag{3}$$

where J_k is a diagonal matrix, $diagonal(\mathbf{p}_k)$ formed from the elements of $\mathbf{p}_k$. Similarly, for equation (2), $P^{(k)}(P^{(k)})^T$ is a symmetric $n \times n$ matrix whose off-diagonal elements correspond to the adjacency matrix of the graph on n nodes having complete graph components for each cluster in the kth partition. Equation (2) becomes

$$S = \sum_{k=1}^{r} P^{(k)}(P^{(k)})^T = \sum_{k=1}^{r} J_k + \sum_{k=1}^{r} A_k \tag{4}$$

where again J_k is a diagonal matrix, but now formed from the elements of kth partition with $J_k = diagonal(P^{(k)}\mathbf{1})$. If the partition matrix $P^{(k)}$ always includes a column for every singleton cluster in the partition, then J_k is simply the $n \times n$ identity matrix and equation (4) is simply

$$S = rI_n + \sum_{k=1}^{r} A_k \tag{5}$$

In all of these cases, the diagonal terms are of no import and we need only the matrix formed from a weighted sum of adjacency matrices. If we further restrict consideration to unity weights, then both equations (3) and (4) lead to a sum of adjacency matrices

$$\sum_{k=1}^{m} A_k \tag{6}$$

for appropriately defined m.

2.2 Using the raw weighted adjacency matrix

The previous section suggested that the sum of the adjacency matrices of equation (6) contains information on the ensemble of the multiple clusterings. To assess this, we consider an example of multiple clusterings produced by different restarts of a k-means algorithm on data from a simple Gaussian mixture of known cluster structure. While this investigation will show that this sum does contain considerable information on the ensemble, it will also show that it also can provide highly variable detail that can hide important cluster structure as well.

2.2.1 Data from a Gaussian mixture

Figure 1 shows a sample 300 points, 100 drawn from each of three distinct two dimensional Gaussians. The circles are drawn from the first Gaussian (Group 1 of Figure 1), the squares from the second (Group 2), and the triangles from the third (Group 3). The first and second Gaussians are located closer to each other than either is to the third. The underlying hierarchy generating the groups is the tree of Figure 2.

Suppose, as in Fred and Jain (2002) or Ashlock, Kim, and Guo (2005), we were to attempt to cluster these data using a partitioning algorithm like k-means. Clearly no hierarchical clustering could result. However, because k-means will only find local optima, random restarts of k-means will typically output different clusterings (even with the same value of k). The ith restart will produce a partition, $P^{(i)}$, and hence graph G_i (having k complete components) with adjacency matrix A_i. With m random restarts, we compute the sum $A_\omega = \sum_{i=1}^{m} A_i$ of these adjacency matrices and then examine A_ω to uncover any consensus clustering pattern.

A simple visual means to display the sum A_ω is as a pixel map, or pixmap. A pixmap can be used to display the contents of the matrix A_ω by replacing each matrix element by a tiny square or pixel (i.e. a 300×300 array of pixels in this case) whose colour saturation is determined by its corresponding value in the total adjacency matrix, A_ω – the larger is the value, the darker is the colour. Diagonal cells are assigned the maximum saturation for display purposes.

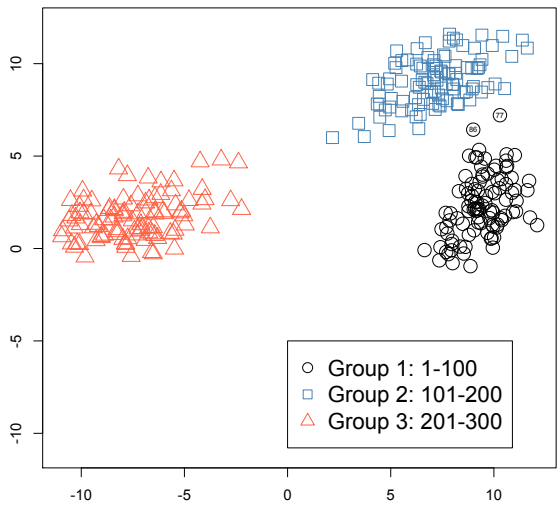


Figure 1: A data set from a mix of Gaussians (points numbered 77 and 86 are identified)

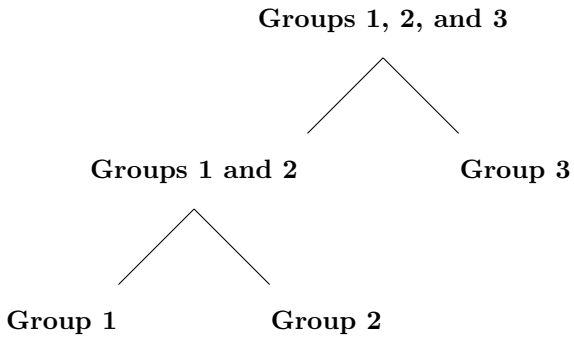
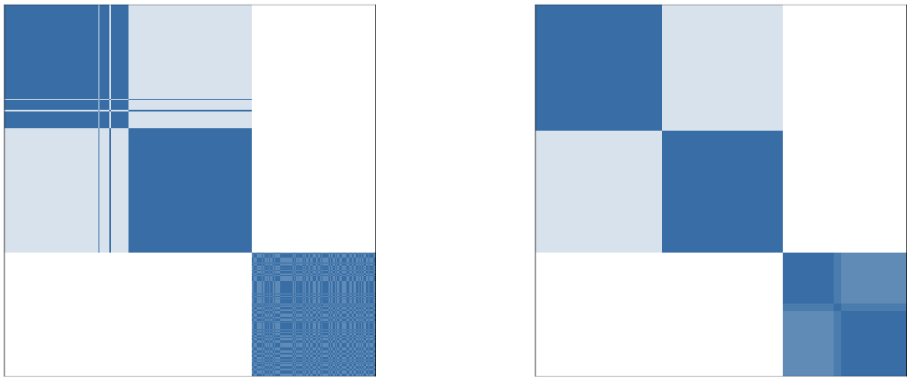


Figure 2: The true hierarchical cluster tree generating the Gaussian data

2.2.2 Clusterings from k-means with k=3 and 10 random restarts

The sum of the adjacency matrices from 10 random restarts of k-means clustering (with $k = 3$ each time) is shown as the pixmap of Figure 3(a) with rows (numbered 1 to 300 from the top) and columns (1 to 300 from the left) matching the indices of the points from Figure 1. The more often a pair of points has been assigned to the same cluster over the 10 restarts, the greater the saturation of colour in the corresponding cell.

The three larger dark squares in Figure 3(a) correspond to the three groups of 100 points from Figure 1 indicating that the different restarts have often placed each group's points within its own cluster. The restarts have also often placed points in group 1 in the same cluster as points from group 2 as shown by the lighter coloured off-diagonal squares where group 1 points are paired with group 2 points. This is because groups 1 and 2 are near each other and are elliptically shaped whereas k-means will often (depending on the restart) mix the two groups to form its preferred circular clusters. This preference for circular clusters also explains the two parallel lines (vertical and horizontal) corresponding to points 77 and 86 (see Figure 1) which were assigned more often to the same cluster as points from group 2 than a cluster of points from



(a) Rows and columns in original data order

(b) Reordered rows and columns

Figure 3: Pixmap plot of $A_\omega = \sum_{i=1}^{10} A_i$ from 10 random restarts of k-means with $k = 3$.

group 1.

The patterns in Figure 3(a) stand out because the columns and rows have been ordered to match the data ordered by group as in Figure 1. Had the row order been random, it would have been difficult to discern the pattern. This happens, for example, within the square of the third group (bottom right square of Figure 3(a)) where a variety of darker and lighter cells appear without pattern.

In Figure 3(b), all 300 rows and columns have been rearranged (symmetrically) so as to have the darkest cells appear nearest the diagonal of the matrix (accomplished here using the `dissplot` function with `method = tsp` from the `seriation` `R` package of Michael Hahsler and Buchta (2008) – see also Bivand et al. (2009)). Such an arrangement makes it much easier to perceive the clustering structure uncovered by the restarts.

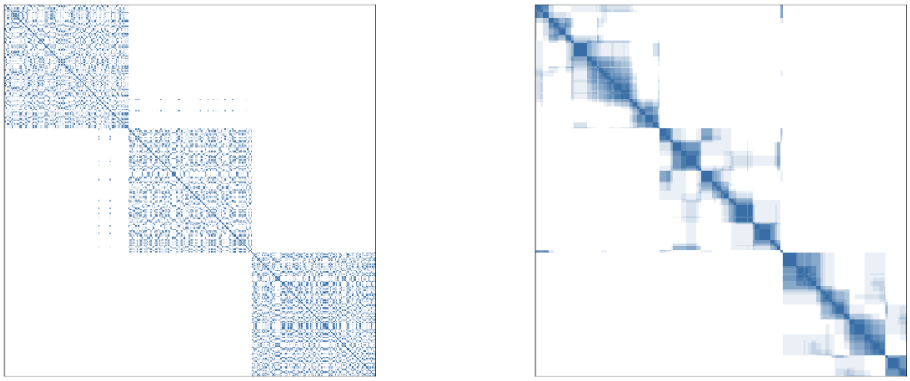
The hierarchy noted in Figure 3(a) is crisper in Figure 3(b) (e.g. rows and columns corresponding to points 77 and 86 now align to group two) and further structure now appears in the third group. Within group 3 three more levels of hierarchy appear (including three clusters at the lowest level). Again, one reason that the k-means restarts are likely asserting this hierarchy is because of the method's predisposition to circular clusters. Note, however, that this additional hierarchy within group 3 is not being asserted as strongly as is that between groups 1, 2, and 3. The evidence for this is the difference in saturation levels within group 3 compared to that between the three groups. Combining the restarts has thus uncovered (mostly correctly) the hierarchy of Figure 2 and has posited some further depth in the cluster tree.

In this example, the sum of adjacency matrices appears to work well to capture the hierarchical structure apparent in Figure 1. Indeed, were one to apply single linkage to the sum A_ω, as if it were a similarity matrix, then the hierarchical clustering as just described would result. This is not always the case.

2.2.3 Clusterings from k-means with k=16 and 10 random restarts

If, instead of the correct value of $k = 3$ for each random restart of k-means, we were to combine the outcomes from random restarts with $k = 16$ then a very different result is obtained for this data. The resulting A_ω of 10 such random restarts is shown in Figure 4(a) with rows and columns in the original order and in Figure 4(b) with order rearranged to have the darkest cells appear close to the diagonal.

The three groups are evident in the sum A_ω as seen in Figure 4, even though the number of clusters given to k-means is incorrect. However, the hierarchical structure so apparent in Figure 3 has disappeared in Figure 4. In its place, finer, more complex, structure is presented within each group (Figure 4(b)). Single



(a) Rows and columns in original data order (b) Reordered rows and columns

Figure 4: Pixmap plot of $A_\omega = \sum_{i=1}^{10} A_i$ from 10 random restarts of k-means with $k = 16$.

linkage applied to this A_ω would fare the same, retaining uninteresting detailed structure.

2.2.4 A need to smooth

As the above two examples show, the sum of the adjacency matrices, A_ω, pools information from the contributing clustering outcomes in a way that seems to capture some of the large scale structure. However, A_ω, being the sum of several adjacency matrices, can also have considerable variability in its elements (as Figure 4 has shown). The effect is easily observed in the pixmap display but would also manifest itself as a bushy dendrogram from say single-linkage. What is needed is some kind of "smoothing" operation on the A_ω which retains the principal clustering structure but which dampens the rest.

In the next section, we develop a formal approach to addressing this problem.

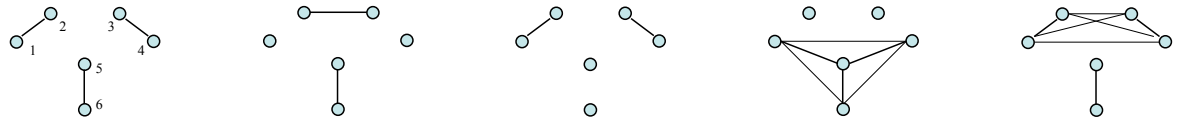
2.3 Developing the methodology on families of graphs

As each adjacency matrix, A_i, corresponds to a graph, G_i, on the n points, we take the existence of a collection of graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_m\}$ and their corresponding adjacency matrices $A_1, \ldots, A_m$ as our starting point. Each $G_i = (V_i, E_i)$ is a *labelled graph* with vertex set $V_i \subset \{1, \ldots, n\}$ representing the n objects and edges in E_i representing some association between the corresponding objects. A single graph could correspond to a partition, a single cluster, or multiple, non-overlapping, clusters. The adjacency matrix, A_i associated with any graph $G_i \in \mathcal{G}$ will be $n \times n$, regardless of the number of vertices in V_i. We call $\mathcal{G}$ a *graph family*.

Our objective will be to turn this collection of graphs into a single, suitably pruned, cluster tree. The approach will be to abstract the steps as we go so as to provide a firm foundation for the process. A single simple example will be used to illustrate.

2.3.1 The starting graph family

For example, consider the graphs of Figure 5. Each represents a partition of six planar data points with a component being a cluster. The corresponding adjacency matrices are shown in Figure 6.



G_1 G_2 G_3 G_4 G_5

Figure 5: A collection of five graphs on a single set of six data points (as labelled in G_1). Each graph component is a cluster. The collection is called a graph family.

$$A_1 = \begin{pmatrix} 0&1&0&0&0&0 \\ 1&0&0&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&0&0&1 \\ 0&0&0&0&1&0 \end{pmatrix} \quad A_2 = \begin{pmatrix} 0&0&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&1 \\ 0&0&0&0&1&0 \end{pmatrix} \quad A_3 = \begin{pmatrix} 0&1&0&0&0&0 \\ 1&0&0&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{pmatrix}$$

$$A_4 = \begin{pmatrix} 0&0&0&1&1&1 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 1&0&0&0&1&1 \\ 1&0&0&1&0&1 \\ 1&0&0&1&1&0 \end{pmatrix} \quad A_5 = \begin{pmatrix} 0&1&1&1&0&0 \\ 1&0&1&1&0&0 \\ 1&1&0&1&0&0 \\ 1&1&1&0&0&0 \\ 0&0&0&0&0&1 \\ 0&0&0&0&1&0 \end{pmatrix}$$

Figure 6: Adjacency matrices corresponding to graphs of Figure 5.

The sum of the adjacency matrices is

$$A_\omega = \sum_{k=1}^{5} A_k = \begin{pmatrix} 0&3&1&2&1&1 \\ 3&0&2&1&0&0 \\ 1&2&0&3&0&0 \\ 2&1&3&0&1&1 \\ 1&0&0&1&0&4 \\ 1&0&0&1&4&0 \end{pmatrix}.$$

The sum A_ω can be thought of as a weighted adjacency matrix for a graph G_ω (edges have non-zero integer weights corresponding to the entries of A_ω). Alternatively, it could be the adjacency matrix recording the multiplicity of each edge in a multigraph. As can be seen, A_ω summarizes all of the partitions in Figure 5 at the expense of some loss of information on the individual graphs.

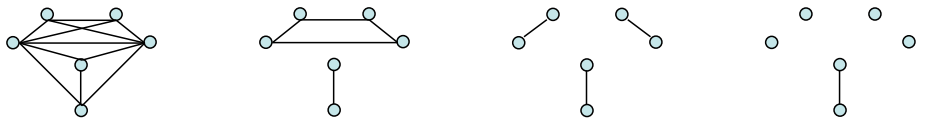
2.3.2 The corresponding nested family

To uncover the hierarchical information within A_ω, note that the same matrix sum results from the set of adjacency matrices shown in Figure 7. These new adjacency matrices, $A_{(i)}$, are ordered in the sense that $A_{(i)} \geq A_{(i+1)}$ for $i = 1, 2, 3$, where "$\geq$" means element-wise "$\geq$" for every pair of elements. As a consequence, the corresponding graphs, $G_{(1)}$, $G_{(2)}$, $G_{(3)}$, and $G_{(4)}$ respectively, shown in Figure 8, are nested and we write $G_{(i)} \succeq G_{(i+1)}$ for $i = 1, 2, 3$ to indicate this nesting. The corresponding partitions are therefore also nested and the sequence $G_{(1)} \succeq G_{(2)} \succeq G_{(3)} \succeq G_{(4)}$ describes a hierarchical clustering. This is the same hierarchy that would result from applying single linkage to A_ω (excluding singletons for those nodes that are nowhere separated in the graphs). Complete linkage could of course be different. The construction of nested graphs, $G_{(1)}$, $G_{(2)}$, $G_{(3)}$, and $G_{(4)}$, is also unique in providing nested partitions whose adjacency matrices sum to A_ω (see Appendix).

From any such monotonic graph family, we may construct a hierarchical clustering of the vertices by following the nested graph components.

$$A_{(1)} = \begin{pmatrix} 0&1&1&1&1&1\\ 1&0&1&1&0&0\\ 1&1&0&1&0&0\\ 1&1&1&0&1&1\\ 1&0&0&1&0&1\\ 1&0&0&1&1&0 \end{pmatrix} \quad A_{(2)} = \begin{pmatrix} 0&1&0&1&0&0\\ 1&0&1&0&0&0\\ 0&1&0&1&0&0\\ 1&0&1&0&0&0\\ 0&0&0&0&0&1\\ 0&0&0&0&1&0 \end{pmatrix} \quad A_{(3)} = \begin{pmatrix} 0&1&0&0&0&0\\ 1&0&0&0&0&0\\ 0&0&0&1&0&0\\ 0&0&1&0&0&0\\ 0&0&0&0&0&1\\ 0&0&0&0&1&0 \end{pmatrix} \quad A_{(4)} = \begin{pmatrix} 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&0\\ 0&0&0&0&0&1\\ 0&0&0&0&1&0 \end{pmatrix}$$

Figure 7: Adjacency matrices corresponding to graphs of Figure 8.



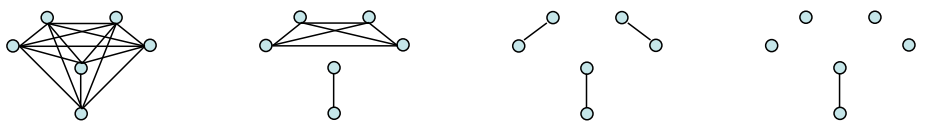
$G_{(1)}$ $G_{(2)}$ $G_{(3)}$ $G_{(4)}$

Figure 8: The nested sequence of graphs $G_{(1)} \succeq G_{(2)} \succeq G_{(3)} \succeq G_{(4)}$. Each graph component is a cluster. The ordered collection is called a monotonic graph family.

2.3.3 Completing graph components

The graphs of $\mathcal{G}_2$ provide a hierarchy of clusters. However, as $G_{(1)}$ and $G_{(2)}$ of Figure 8 show, the corresponding adjacency matrices will have greater variation in their elements than is necessary for, or indicated by, the clustering. Were we to apply single linkage, for example, to the sum of the adjacency matrices of $\mathcal{G}_2$ we would not return the nested sequence of components we see visually in Figure 8.

This variation can be simply reduced by adding sufficient edges so that each component of every graph in $\mathcal{G}_2$ is itself a complete (sub)graph. That is, for clustering structure it makes more sense to work with the graphs of $\mathcal{G}_3 = \{G^*_{(1)}, G^*_{(2)}, G^*_{(3)}, G^*_{(4)}\}$ shown in Figure 9 than those of Figure 8.



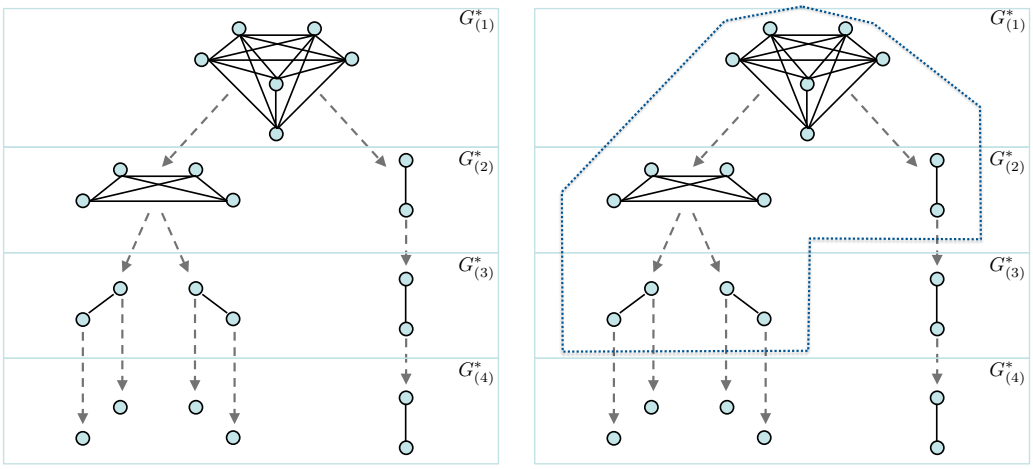
$G^*_{(1)}$ $G^*_{(2)}$ $G^*_{(3)}$ $G^*_{(4)}$

Figure 9: The nested sequence of graphs $G^*_{(1)} \succeq G^*_{(2)} \succeq G^*_{(3)} \succeq G^*_{(4)}$. Each graph component is now complete and corresponds to a cluster. The ordered collection remains a monotonic graph family.

As it happens, the example of Figure 9 includes the complete graph on all vertices. Should it not, and since we are interested in ultimately producing a cluster tree, it would make sense to add the complete graph to the set so as to have a cluster of all vertices for the tree's root node. As noted in Carroll and Corter (1995, pp. 286-7), this is not uncommon and amounts to adding a constant matrix to the right hand side of equation (1), the `ADCLUS` equation.

2.3.4 The component tree

Having arrived at a completed monotonic family of graphs $\mathcal{G}_3$ (as in Figure 9), a tree is easily constructed by nesting all components in the family – the *component tree*. Figure 10(a) illustrates the component tree



(a) Component tree (b) Subtree of interest

Figure 10: Graphs in $\mathcal{G}_3$ layered in nested order with subgraph components connected to form a component tree.

of the graph family $\mathcal{G}_3$ of Figure 9. The component tree exists for any monotonic graph family $\mathcal{G}$ provided the first element is a complete graph (the root) and its component tree can be constructed in this way. The graph families $\mathcal{G}$ of interest here will also have each element consist of complete subgraphs.

For clustering purposes, however, the component tree often contains redundant and uninteresting information and is an unsatisfactory summary of the cluster structure. For this example, the subset of components highlighted in Figure 10(b) is a better summary of the clustering structure. Note that this excludes potentially many components of the component tree.

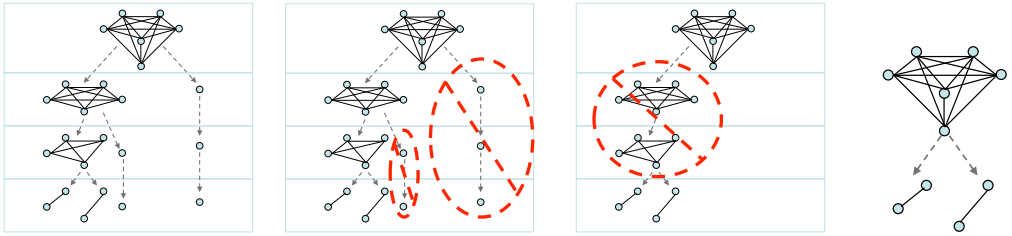
2.3.5 Tree reduction

Beginning with the component tree (or equivalently the graph family on which it is based) we would like to reduce the tree to one that contains only those components that capture the essential cluster structure. To that end, two simple rules are applied to a component tree:

1. Prune the trivial components.
2. Contract pure telescopes of components.

These are applied in order. The resulting component tree will be called the cluster tree.

These rules are illustrated in Figure 11 using a slightly more complicated monotonic graph family than



1. Component tree 2. Trivial pruning 3. Telescopic contraction 4. Cluster tree

Figure 11: Steps in reducing the component tree to a cluster tree.

that of $\mathcal{G}_3$.

Rule one prunes all trivial components from the component tree (step 2 of Figure 11). Here, and in the rest of the paper, single node subgraphs are taken to be "trivial components". Note, however, that a different choice could be made such as all components with m nodes or fewer without affecting any of the discussion. For example, for some clustering problems m might be defined as a proportion of the total number n of objects being clustered so that for large n only large clusters would survive the tree reduction.

Reduction rule two collapses all nested sequences of components where there is no branching. Applied after the first rule, this rule ensures that only branching into two or more non-trivial components is preserved in the cluster tree.

Application of these two rules reduces the component tree to a cluster tree. It is important to note that the cluster tree here is not a dendrogram; it shows no singleton nodes (more generally, no trivial nodes). Instead, like a high-density cluster tree detecting modes in a density, principal interest lies in detecting when groups occur and separate, as well as in the contents of each group at the point of separation (e.g. see Stuetzle 2003).

Applying the rules to the component tree of Figure 10(a) will yield the subtree outlined in Figure 10(b) as the cluster tree for graph family $\mathcal{G}_3$.

The cluster tree can itself now be summarized by the sum of the $n \times n$ adjacency matrices of the graphs that remain in the cluster tree. Denote this matrix by A_ρ, where ρ is chosen to alliteratively suggest the reduction step just completed. For the cluster tree of $\mathcal{G}_3$, this sum is

$$A_\rho = \begin{pmatrix} 0 & 3 & 2 & 2 & 1 & 1 \\ 3 & 0 & 2 & 2 & 1 & 1 \\ 2 & 2 & 0 & 3 & 1 & 1 \\ 2 & 2 & 3 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 2 \\ 1 & 1 & 1 & 1 & 2 & 0 \end{pmatrix}. \tag{7}$$

That this summarizes the structure of the cluster tree appearing within the dotted lines of Figure 10(b) is easily seen from its corresponding pixmap saturation matrix, say S_ρ, with

$$S_\rho = \begin{pmatrix} 2 & 2 & 1 & 1 & 0 & 0 \\ 2 & 2 & 1 & 1 & 0 & 0 \\ 1 & 1 & 2 & 2 & 0 & 0 \\ 1 & 1 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 2 \end{pmatrix} \tag{8}$$

which removes the root node and puts maximum saturation (here 2) on each diagonal element. As constructed, A_ρ and S_ρ roughly correspond to equations (6) and (5), respectively (with suitable adjustment for the universal cluster, or root node, $\mathbf{1}\mathbf{1}^T$).

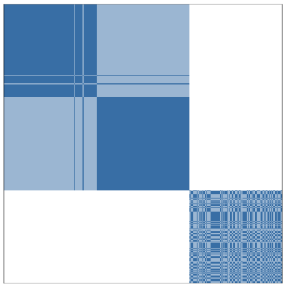
We call the entire method (leading to A_ρ of equation (7)) *tree-reduced ensemble clustering* (or *TREC*).

2.4 Application to multiple clusterings

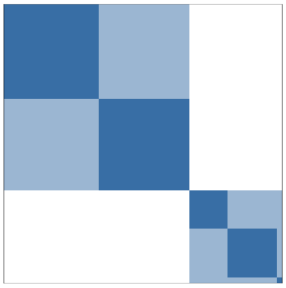
We now apply TREC to some examples of multiple clusterings. First, we consider again the combination of the clusterings from random restarts of k-means for fixed k on the Gaussian mixture data of Section 2.4. TREC methodology, however, is completely agnostic on the methods used to produce the clusterings – TREC will combine any number of clusterings from any number of methods. To illustrate this, we consider a few different methods of clustering the Gaussian mixture data of Figure 1 and show the A_ρ pixmap that results from TREC.

2.4.1 Back to the Gaussian mixture

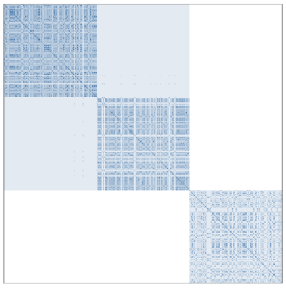
Recall the data from the Gaussian mixture (shown in Figure 1). Figure 12 shows the pixmap representation



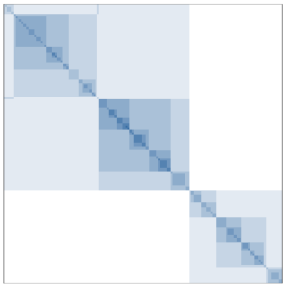
(a) $k = 3$ – rows and columns in original data order



(b) $k = 3$ – reordered rows and columns



(c) $k = 16$ – rows and columns in original data order



(d) $k = 16$ – reordered rows and columns

Figure 12: Pixmaps of A_ρ for TREC on k-means with ten random restarts.

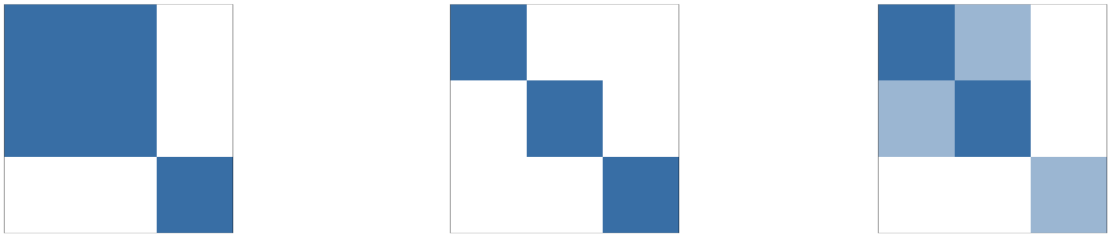
of the A_ρ that results from applying TREC to the graph families for each set of ten random restart k-means clusterings. In Figure 12 we have $k = 3$ for (a) and (b) (original order and reordered for presentation, respectively) and $k = 3$ for (c) and (d) (original and optimized presentation order, respectively).

For $k = 3$, comparing TREC's A_ρ of Figures 12(a) and (b) with the adjacency matrices A_ω of Figures 3(a) and (b), we see some similarity. In both sets of Figures, there are two major clusters, and then within each of these there is further separation into subgroups. Both describe groups 1 and 2 the same (including points 77 and 86). However, TREC's A_ρ has made k means assertion of further hierarchy within the third group (bottom right corner) simpler and more apparent. Note also that the layers of the pixmap for TREC's A_ρ are more noticeable because there are fewer layers in the corresponding cluster tree and colour saturation follows the cluster tree layer level for each group.

In contrast, for $k = 16$ there is a good deal of difference between TREC's A_ρ and the raw adjacency matrix A_ω. Comparing Figures 12(c) and (d) with the Figures 4(a) and (b), respectively, we see that A_ρ presents a simpler and more hierarchical pattern than does A_ω. TREC has, with $k = 16$, identified the hierarchy which previously had only been discovered with $k = 3$. Groups 1 and 2 appear separately from one another and together as a group separately from group 3 in Figure 12(d). Further, additional subgrouping can be seen within each of the three groups, reflecting the local density that has been picked up by the large value of $k = 16$.

2.4.2 Combining k-means and model-based clustering

One instance of k-means clustering with $k = 2$ is used to produce the adjacency matrix shown in the pixmap of Figure 13(a). In contrast, model based clustering (e.g. Fraley and Raftery 1999), based on a mixture



(a) k-means ($k = 2$), no restarts (b) Gaussian mixture model-based clustering (c) TREC combination

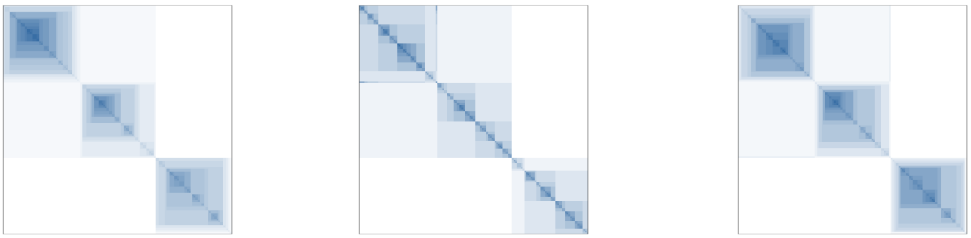
Figure 13: Pixmap plots of A_ρ from TREC on two different partition methods

of Gaussian distributions, found three clusters as shown in Figure 13(b). Their combination, by TREC, produces the expected hierarchy shown by the pixmap of A_ρ in Figure 13(c).

2.4.3 Combining single- and complete-linkage clusterings

Clusterings to be combined need not be partitions. Since every hierarchical clustering method is itself a set of nested partitions (or, equivalently, a monotonic graph family), TREC can be applied to each hierarchy by itself, or to any combination of hierarchies.

In particular, if we apply TREC to the results of single linkage on the Gaussian mixture data, the resulting adjacency matrix A_ρ is given by the pixmap of Figure 14(a). Similarly, applying TREC to the results of



(a) TREC on single linkage (b) TREC on complete linkage (c) TREC on the two together

Figure 14: Pixmap plots of A_ρ from TREC on two different hierarchical methods

complete linkage, also a hierarchical method, produces the A_ρ illustrated in Figure 14(b).

Oftentimes, it is difficult to choose between single and complete linkage and which is to be preferred has been a matter of some debate historically (e.g. see Shepard and Arabie 1979, page 90). In Figure 14(a) and (b), both reveals the gross hierarchical structure and the three groups. In Figure 14(b), we see that complete linkage asserts more detailed structure within each group (like k-means complete linkage prefers tight globular clusters). Figure 14(c) shows the TREC combination of single and complete linkage. While a smooth blending of the two, Figure 14(c) appears to be closer to single linkage's Figure 14(a) than complete

linkage's Figure 14(b). This is consistent with TREC's tendency to reduce the number of levels in the cluster tree.

2.4.4 A spiral data set

Many more examples could be constructed to show the value of tree reduced ensemble clustering. We conclude this section with an instance of spiral clusters and k-means clustering. Figure 15 shows 100 data

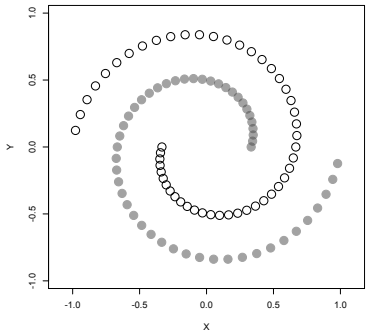


Figure 15: Two spiral groups: 50 points in each group.

points, 50 in each of two separate but interlocked spirals.

Figure 16(a) shows the pixmap that results when TREC is applied to ten random restarts of k-means, for $k = 2$. One group is labelled sequentially from 1 to 50, and the second group from 51 to 100; the rows and columns of the pixmap are ordered using these labels. Although $k = 2$ is the correct number of clusters, the combination of several k-means partitions for this value fails to find the two groups. This is clear from the pixmap of Figure 16(a) which shows the TREC hierarchical structure being cut up across the two groups.

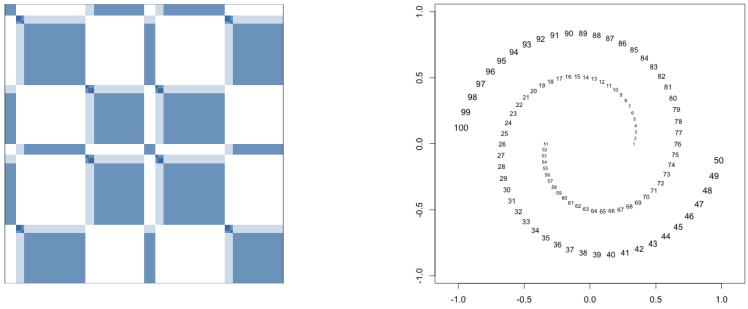
Figure 17(a) rearranges the rows and columns of the pixmap so that the hierarchical structure found by TREC is revealed. The points in Figure 17(b) are relabelled with the row (column) numbers where they appear in the rearranged pixmap. Again, Figures 17(a) and (b) show that k means, for $k = 2$, is incapable of recovering these two groups.

Even so, the hierarchical structure in the pixmap can be matched to the labelled points to see how TREC gathers information from several single k-means clusterings with $k = 2$. From top-left to bottom-right, the nested squares of the pixmap have the following ranges of row and column numbers: $\{\{1-44\}, \{\{45,46\}, \{47-50\}\}\}$, $\{\{51-94\}, \{\{95-98\}, \{99,100\}\}\}$. The two largest blocks ($\{1-50\}$ and $\{51-100\}$) split the data points in two, roughly along a 30 degree line in Figure 17(b). The two largest coloured blocks in the pixmap appear on either side of this line.

In the present example, the problem is that the local structure is difficult to capture for k-means when $k = 2$. To be sensitive to global structures which are not globular, k should be chosen to be much larger than the number of groups expected. One approach, then, is to choose a single large value of k and use several random restarts; another is to choose many values of k over a large range, say from $k = 2$ to $k = 32$.

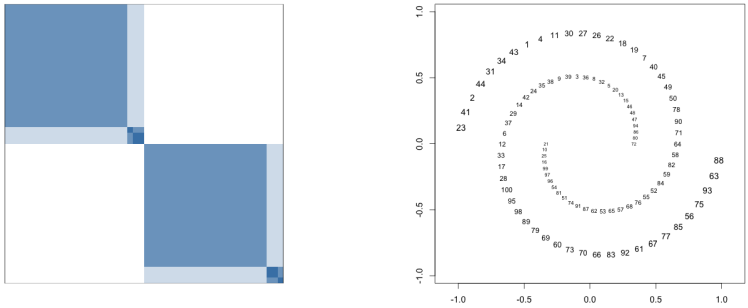
Figure 18(a) shows the pixmap which resulted from ten random restarts of k-means on this data for $k = 20$. The order of the rows (and columns) match the true order of the data points as shown by the labelled points of Figure 16(b). As this pixmap reveals, TREC is able to correctly uncover the two clusters.

Varying k from $k = 2$ to $k = 32$ and applying TREC to the 31 resulting partitions yields the pixmap shown in Figure 18(b). Again the row (column) order is as in Figure 16(b) and the pixmap shows the two spiral groups as separate. In both Figures 18(a) and (b), the k-means partitions also identify some local structure within each group.



(a) TREC on k=2, 10 restarts. (b) Points labelled by row numbers in (a).

Figure 16: Pixmap plots of A_ρ from TREC on k-means (k = 2) for data shown in Figure 15. In (a), the row order is as shown in (b).



(a) TREC on k=2, 10 restarts. (b) Points labelled by row numbers in (a).

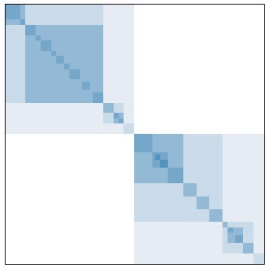
Figure 17: Pixmap plots of A_ρ from TREC on k-means (k = 2) for data shown in Figure 15. In (a), the row order is as shown in (b).

3 Formalizing the graph algebraic framework

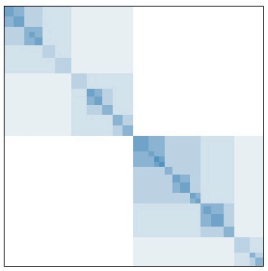
Graphs have long been a natural way to represent clusters and have motivated a number of methods (e.g. Godehardt 1990; Sneath and Sokal 1973). Throughout Section 2, this relationship was explored and exploited to summarize collections of such graphs in a single cluster tree. The discussion of graphs and graph families led to the methodology of tree reduced ensemble clustering or TREC (i.e., up to and including equation 6).

Following the reasoning of Section 2.3, it is easy to see that the TREC methodology is simply a series of transformations on an arbitrary collection of graphs which outputs a cluster tree. This sequence of transformation can be mathematically formalized so that the transformations, and their relations to one another are easily understood. Perhaps more significantly, these relationships establish a graph algebraic framework that provides a formal means to map *any* clustering method to a cluster tree.

By abstracting this framework, an isomorphism between the set of such cluster trees and a space of weighted adjacency matrices can be established. A metric on the latter space is easily had, one which, as a consequence of the isomorphism, is also seen to provide a reasonable *measure of distance between any two cluster trees*. This distance measure can then be used to compare clusterings in any observational or experimental setting.



(a) TREC on k-means ($k = 20$), 10 random restarts



(b) TREC on k-means ($k \in \{2, 3, \ldots, 32\}$)

Figure 18: Pixmap plots of A_ρ from TREC on different settings of k-means. The first spiral appears in the first 50 rows (columns), the second in the last 50. Row order within each spiral is counter clockwise.

In this section, we abstract these transformations in Section 3.1, describe the relations between the sets on which they operate in Section 3.1.1, and describe some propertied in Section 3.1.2. Proofs of all properties are relegated to the Appendix.

Because of the isomorphisms described in Section 3.1.2, we are able in Section 3.2 to define a distance function on the set of adjacency matrices whose elements are the possible A_ρs which is able to serve as a distance between the corresponding graph families and hence their unique cluster trees.

In Section 3.2.1 we illustrate the application of this new distance between cluster trees by examining the distances between all clusterings proposed for the Gaussian mixture data of Section 2.2.1. The examples show that this distance provides a meaningful measure with which various clustering outcomes can be prepared. TREC is also seen to behave favourably in the sense that it is nearly in every instance closer to the true tree for the Gaussian mixture than are any of the clustering outcomes on which it is based.

3.1 The framework

The basic element of the framework is a *graph family*, denoted $\mathcal{G}$, which is a collection of graphs with identifiable nodes (in that nodes are determinably either identical or different across graphs). A simple example is $\mathcal{G} = \{$  $\}$ from Figure 5 of Section 2.3.1 – here all nodes are common across graphs. A graph family is not necessarily a set, in that graphs may appear more than once within some $\mathcal{G}$.

From Section 2.3.1, we can abstract a sum operator, ω, which, when applied to a graph family $\mathcal{G}$, will yield the *weighted* adjacency matrix $A_\omega = \omega(\mathcal{G})$ that is the sum of the adjacency matrices of all graphs in $\mathcal{G}$.

In Section 2.3.2, the weighted adjacency matrix is used to construct a new graph family, $\mathcal{G}_M$ say, this one a monotonic graph family in that its graphs form a nested sequence. Let γ denote this *graph* family producing operation so that $\mathcal{G}_M = \gamma(A_\omega)$. The composition of $\gamma \circ \omega$ takes any graph family $\mathcal{G}$ to a monotonic graph family $\mathcal{G}_M$. For some sets of graph families and of weighted adjacency matrices, γ and ω are inverses of one another (see the Appendix for details).

In Section 2.3.3, we ensure that the monotonic graph family $\mathcal{G}_M$ is completed in the sense that every graph component is a complete subgraph and the family includes the complete graph on all n nodes. Let κ denote this *completion* operation on $\mathcal{G}_M$.

From Section 2.3.4, we can abstract a *tree* constructor, say τ, that constructs the component tree from a monotonic graph family containing the complete graph on all nodes.

In Section 2.3.5, a monotonic family is reduced by employing reduction rules on its component tree. Denote this *reduction* operator by ρ which applies to a completed monotonic graph family and returns a new monotonic graph family (containing no trivial or cluster uninformative components) according to the reduction rules (expressed in terms of the families' component trees).

The TREC procedure, therefore, simply maps any graph family, $\mathcal{G}$, to a cluster tree, T_{clus}, via the

composition

$$T_{clus} = \tau \circ \rho \circ \kappa \circ \gamma \circ \omega(\mathcal{G}).$$

TREC is entirely agnostic about how the family of graphs $\mathcal{G}$ is constructed or otherwise obtained and so applies to any graph family representation of a clustering.

3.1.1 Relating the sets

With the above operations in place, it becomes of interest to describe their domains and ranges in terms of the various sets of graph families, their isomorphic sets of weighted adjacency matrices, and the mappings from one set of graph families to another which lead ultimately to a cluster tree.

These are connected by considering the carefully prescribed sequence of operations on a graph family that ultimately leads to a cluster tree, T_{clus}. The operation sequence which defines TREC follows the path along the bottom row of Figure 19.

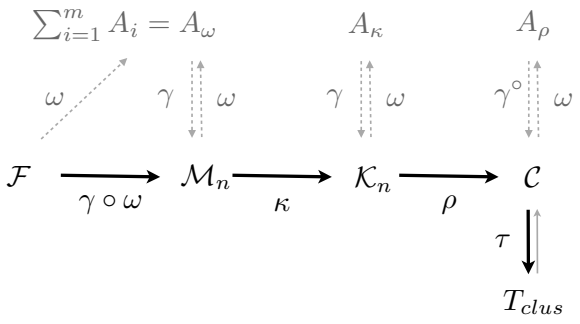


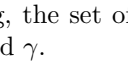
Figure 19: A graph algebraic framework for clustering. Tree reduced ensemble clustering follows the black arrow path across the bottom – from an arbitrary graph family $\mathcal{F}$ to the cluster tree T_{clus}.

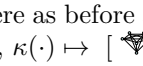
Here, $\mathcal{F}$ represents the set of all possible graph families on n objects (which label the vertices). In our working example from Section 2, $n = 6$ and the graph family $\mathcal{G} =\{$  $\}$ from Figure 5 is an element of $\mathcal{F}$.

The composition $\gamma \circ \omega$ projects $\mathcal{G}$ into $\mathcal{M}_n$, a set of monotonic graph families on n distinct nodes. It does this by first ω mapping each graph family $\mathcal{G}$ in $\mathcal{F}$ to a unique element (A_ω) in a space of weighted adjacency matrices (a many to one mapping) and then having γ map the weighted adjacency matrix to a unique monotonic graph family in $\mathcal{M}_n$.

The subscript n is used here to distinguish the set of all monotonic graph families on n objects, $\mathcal{M}$, from its proper subset, $\mathcal{M}_n$. Graph families of $\mathcal{M}_n$ contain only graphs whose vertex sets are the entire set of n objects $\{1, \ldots, n\}$. In contrast, a family in $\mathcal{M}$ may contain a graph whose vertex set is a proper subset of $\{1, \ldots, n\}$. In our example, $\gamma \circ \omega$ applied to $\mathcal{G}$ yields [] $\in \mathcal{M}_6$ from Figure 8 (note the four singletons in the last graph as required by $\mathcal{M}_6$).

The composition is idempotent with $\mathcal{M}_n \subseteq \mathcal{F}$ and so can be regarded as a projection operator from $\mathcal{F}$ onto $\mathcal{M}_n$. While the composition is a many to one mapping, the set of weighted adjacency matrices (e.g. A_ω) is isomorphic to $\mathcal{M}_n$ through the inverse operators ω and γ.

Similarly, the operator κ projects $\mathcal{M}_n$ onto $\mathcal{K}_n \subset \mathcal{M}_n$, this time simply by completing the components of every graph in a family in $\mathcal{M}_n$. The subscript n is used here as before and we note in passing that κ could be used to directly project $\mathcal{M}$ onto $\mathcal{K} \subset \mathcal{M}$. In the example, $\kappa(\cdot) \mapsto$ [] $\in \mathcal{K}_6$ from Figure 9. By completing components, κ ensures that every vertex in a component contributes equally to that cluster.

The reduction operator ρ projects $\mathcal{K}_n$ onto $\mathcal{C} \subset \mathcal{K}$ by removing cluster irrelevant components, retaining component splits only when two or more non-trivial components result. Note that $\mathcal{C} \not\subset \mathcal{K}_n$, since members of $\mathcal{C}$ may contain graphs with fewer than n vertices. In the example, $\rho(\cdot) \mapsto$ [] $\in \mathcal{C}$, matches the components of the cluster tree highlighted in Figure 10(b).

The final operation, $\tau(\cdot)$, simply produces the cluster tree (e.g. as in Figure 10(a)) from a family of graphs in $\mathcal{C}$. This operation is easily defined to be bijective between $\mathcal{C}$ and the set of all possible cluster trees on n objects.

3.1.2 Isomorphic relations

As Figure 19 suggests, the same operators, γ and ω, used to project any graph family in $\mathcal{F}$ onto a member of the set of monotonic graph families $\mathcal{M}_n$ can also be used to isomorphically move between a graph family representation and its weighted adjacency matrix representation. (See the Appendix for proofs.)

In particular,

- an A_ω uniquely matches each single $\mathcal{G} \in \mathcal{M}_n$,
- an A_κ uniquely matches each single $\mathcal{G} \in \mathcal{K}_n$, and
- an A_ρ uniquely matches each single $\mathcal{G} \in \mathcal{C}$.

In the last of these a slight adjustment to γ is made to no longer insist on single vertices being added – γ° is γ with omission of single vertices.

Similarly, τ maps any graph family in $\mathcal{C}$ to a unique cluster tree T_{clus}. Moreover, τ is invertible. As the last column of Figure 19 suggests, this implies that we have isomorphic relationships between weighted adjacency matrices A_ρ and cluster trees T_{clus} (via some graph family in $\mathcal{C}$).

These isomorphisms allow operations between members of one set to be defined and explored in terms of operations between members of any other set. We simply choose which of the three sets (weighted adjacency matrices, graph families, or cluster trees) is most amenable to the operation.

This property was implicitly exercised throughout Section 2, where a traditional (absent singletons) tree layout is motivated by T_{clus} and the pixmap layout for the equivalent A_ρ.

3.2 A distance between cluster trees

A measure of similarity, or dissimilarity, between clusterings is of potential value in cluster analyis – for example, to assess qualities of competing clusterings on a particular dataset or to measure performance of different clustering methods in an experimental setting. An important consequence of the algebraic framework of Figure 19 is that, provided the clusterings are expressed as cluster trees, a distance measure between clusterings can be defined via the weighted adjacency matrices A_ρ.

All of the information of a cluster tree, T_{clus} is contained in the upper triangle (excluding the diagonal) of the corresponding weighted adjacency matrix A_ρ or, equivalently, of its saturation matrix S_ρ (e.g. see equations 7 and 8). For simplicity of presentation, we take the upper triangle of the saturation matrix S_ρ and represent it as a vector $\mathbf{s}_\rho$, say, having $\binom{n}{2}$ elements and formed by stacking the columns of the upper triangle of S_ρ on top of one another (the standard "vec" operation applied only to the upper triangle).

Suppose that we have two clustering trees, T_1 and T_2, on the same data and we let the corresponding $\mathbf{s}_\rho$ for each be $\mathbf{s_1}$ and $\mathbf{s_2}$, respectively. We define a distance $d(T_1, T_2)$ to be

$$d(T_1, T_2) = \left|\left| \frac{\mathbf{s_1}}{||\mathbf{s_1}||} - \frac{\mathbf{s_2}}{||\mathbf{s_2}||} \right|\right|,$$

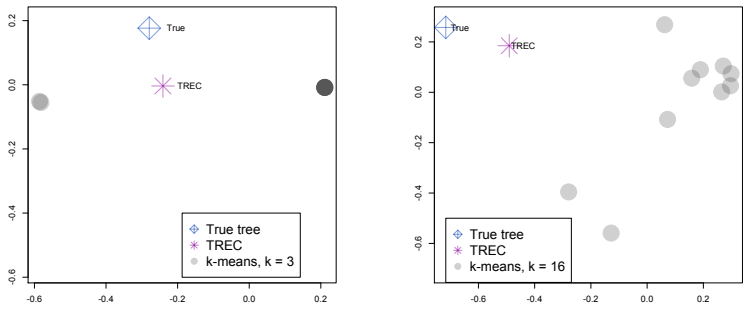
where $||\cdot||$ denotes the Euclidean norm. This definition is not applicable when either of $\mathbf{s_1}$ or $\mathbf{s_2}$ has Euclidean length zero – a trivial restriction as this occurs only when the corresponding cluster tree is "trivial" in the sense of consisting only of the root node. In all other cases, this is provably a metric on the set of cluster trees (e.g. see Zhou (2010)). The distance has been explored in some detail in Zhou (2010) where it is shown to have many desirable properties as a distance between cluster trees.

Following Figure 19, any clustering method can be mapped (via TREC) to an appropriate summary adjacency matrix A_ρ. This in turn is isomorphic to a cluster tree. The distance between different outcomes and/or methods can then be measured and used to place the trees in a Euclidean space via, for example, a method such as multidimensional scaling.

3.2.1 The Gaussian mixture results revisited

For the Gaussian mixture example, the true tree is known (see Figure 2) and following Figure 19 we see that each clustering outcome can be mapped to a cluster tree. These cluster trees are estimates of the true tree and some sense of their quality can be hand calculating their distance to the true tree as well as to one another. For every pair of clustering outcomes (i, j) in Section 2.3, the distance $d(T_i, T_j)$ between their corresponding cluster trees is calculated as is each tree's distance to the true tree $d(T_i, T_{true})$. These distances (or subsets of them, depending on the focus) are then input to a multidimensional scaling algorithm which will assign a two-dimensional position to each tree.

Figure 20(a) shows the result of this for the 10 restarts of k-means for $k = 3$, their TREC combination,



(a) k=3, 10 restarts. (b) k=16, 10 restarts.

Figure 20: Cluster tree results for the Gaussian mixture data. Multidimensional scaling on distances between trees has been used to provide cluster tree coordinates.

and the true tree. The points corresponding to k-means are shown as grey filled circles. The colours have alpha blending so that the over-striking of points show as darker grey. The restarts only effectively produced two cluster trees – one at the left of Figure 20(a) with a small number of points overstriking and one at the right with many more points overstriking (a much darker grey). The aspect ratio of all of these plots is 1 so that the visual distance between points corresponds to the determined distance. Note that the restarts here produced trees on either side of the true tree and that TREC tree was in this instance closer to the true tree than any of the restarts.

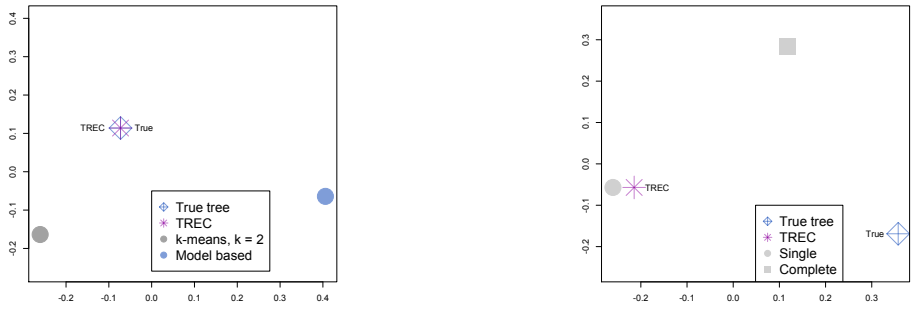
Figure 20(b) shows the case of 10 restarts of k-means with $k = 16$. Here all 10 restarts produced different trees and there is little in the way of overstriking. The TREC tree is considerably closer to the true tree than is any of the restarts. The plot also shows clearly that the TREC process does not result in an average position of its tree with respect to those it has combined.

Figure 21(a) shows the placement of the trees related to combining model based clustering with k-means ($k = 2$). Here the TREC combination exactly reproduced the true tree, having distance zero from it.

In Figure 21(b) on the other hand, the combination of single and complete linkage (after processing according to Figure 19) produced a TREC tree which was closer to the true tree than the (TREC reduced) single linkage tree but farther away than the (TREC reduced) complete linkage tree.

In order to make comparisons across all of these methods, all distances must be simultaneously input to the multidimensional scaling algorithm. This was carried out with the positions for all trees as shown in Figure 22. Because so many distances have to be considered in the dimensionality reduction, the relative positioning of points are subject to some minor change.

A number of observations on clustering methods may be had from examining Figure 22. With respect to k means and multiple restarts, for this dataset the smaller (correct) value of $k = 3$ produced trees closer to the truth than did those with $k = 16$. When $k = 2$ (no restarts), the resulting cluster tree was better than any $k = 3$ or model based clustering. The reason for this is that the distance measure gives greater



(a) Combining model based clustering with $k = 2$ means. (b) Combining single and complete linkage.

Figure 21: Cluster tree results for the Gaussian mixture data. Multidimensional scaling on distances between trees has been used to provide cluster tree coordinates.

weight to differences between trees that occur at layers closest to the root node. For this dataset, the true tree is defined in Figure 2 and has two layers, the first of which is a binary split to the data. Any cluster tree that got this first split right (viz. k-means with $k = 2$) would therefore likely be closer to the true tree than would a tree with three branches at the first layer (e.g. k-means with $k = 3$ or the result from model based clustering seen in Figure 13(b)). We also see that the TREC rescue tree of complete linkage is closer to the true tree than is the TREC reduced tree of single linkage.

With the exception of combining the linkage methods, for this dataset the TREC combined cluster trees were closer to the true tree than any of the trees that went into the combination. For the linkage methods, TREC was better than complete linkage but single linkage was marginally closer of the true tree. Finally, whatever the combination, the TREC method produced cluster trees that are amongst the closest to the true tree for this dataset.

4 Concluding remarks

By considering multiple clusterings, we are led to an emphasis on adjacency matrices and their corresponding graphs. The synthesis of these graphs to produce an overall summary of the clusterings, naturally led to consideration of families of graphs in general, and monotonic graph families in particular. Identifying clusters with complete graph components and then highlighting when graph components split into non-trivial subgraphs led us naturally to a cluster tree. Each of these concepts are laid out in the graph algebraic framework of Section 3.

The framework strongly links (through invertible functions), the cluster tree T_{clus} to the adjacency matrix A_ρ. Either one provides a simple hierarchical structure, displayed as a tree or a pixmap respectively, of the synthesis of all clusterings involved.

The framework provides a means through which the outcomes of any clustering methods may be understood, individually or in concert. As the examples in Section 2 illustrated, partitions, hierarchies, individual clusters, or any combination of these can be placed in the framework and studied. Moreover, the synthesis itself may be regarded as a new method of tree reduced ensemble clustering, or TREC. As was shown, this method is agnostic to how the input clustering information was originally produced. It will take advantage of the variety of clusterings presented, and generally, in our experience, tends to give greater insight about the underlying data. However, actual performance clearly must depend on the quality of clusterings presented to TREC.

Finally, like high-density clustering, the summary product of this graph algebraic approach, is a cluster tree. However, unlike high-density clustering, there is no requirement that the objects to be clustered hold

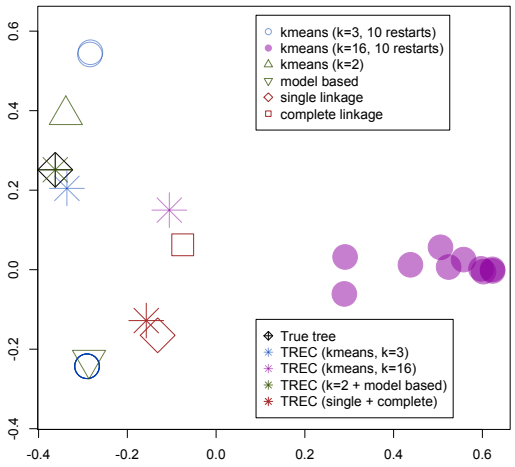


Figure 22: Cluster tree results for the Gaussian mixture data. Multidimensional scaling on distances between trees has been used to provide cluster tree coordinates.

any position in any normed vector space.

This graph algebraic approach, and its attendant ensemble method, provide a framework in which clustering may be cast and methods studied and compared. In particular, the distance between cluster trees introduced in Section 3.2 has considerable promise for investigating the quality of clustering methods. This would be natural in experimental situations where the true clustering is known as was done in Section 3.2.1. Even in observational situations where the true clustering is not known, the distance could be used in conjunction with Euclidean positioning method such as multidimensional scaling (again as in Section 3.2.1) to assess the consistency of various clustering methods applied to the same data. This has been exploited with promising results in Zhou (2010) and has already led to some interesting assessments of clustering methods.

Appendix: Invertible operators, γ and ω.

In Section 3, we assert that the operators γ and ω are inverse functions between certain weighted adjacency matrices and members of certain monotonic graph families. In particular, we consider γ as mapping A_ω onto a member of the set $\mathcal{M}_n$ and conversely ω as mapping a member of the set $\mathcal{M}_n$ onto a weighted adjacency matrix. Here, we prove that, in this case, these mappings are inverses of one another. All other cases shown in Figure 19 will follow as corollaries from this first case.

First we give these operators, γ and ω, precise definitions. This is straightforward for $\omega(\mathcal{G})$ which, as in equation (6), is simply defined to be the "total" of a graph family $\mathcal{G} = \{G_1, G_2, \ldots, G_m\}$, or the graph sum, $G_\omega = \sum_{k=1}^{m} G_k$. Equivalently, either G_ω or its weighted adjacency matrix, A_ω, can be taken as the result $\omega(\mathcal{G})$. With some abuse of notation, we will use these interchangeably in what follows.

Defining γ, however, is a little more involved. Given any weighted graph G_ω with weighted adjacency matrix A_ω having only non-negative integer weights, we construct a monotonic family of graphs, say $\gamma(G_\omega) = [G_{(1)}, \ldots, G_{(r)}]$ such that the corresponding adjacency matrices $A_{(1)} \geq A_{(2)} \geq \cdots \geq A_{(r)}$ sum to A_ω and $r > 0$ is the maximum entry in A_ω.

The construction is as follows. Let $I_{[k,\infty)}(A_\omega)$ be a matrix valued indicator function of its matrix argument, A_ω, which returns a matrix of the same dimensions but having value of 1 in position (i, j) if

the corresponding element of A_ω, w_{ij} say, is greater than or equal to k and zero otherwise. When entries of A_ω are non-negative integers with a finite maximum value of $r > 0$, we define $A_{(k)} = I_{[k,\infty)}(A_\omega)$ for $k = 1, 2, \ldots, r$. We take $G_{(k)}$ to be the graph with n vertices and adjacency matrix $A_{(k)}$. Clearly, $A_{(1)} \geq A_{(2)} \geq \cdots \geq A_{(r)}$ and it follows that $G_{(1)} \succeq G_{(2)} \succeq \cdots \succeq G_{(r)}$. Moreover, $\sum_{k=1}^{r} A_{(k)} = A_\omega$, and so we define $\gamma(A_\omega) = [G_{(1)}, \ldots, G_{(r)}]$.

Finally, let $\mathcal{A}_\omega$ denote the set of all possible weighted adjacency matrices A_ω on n vertices with non-negative integer weights. We may now write $\gamma : \mathcal{A}_\omega \to \mathcal{M}_n$ and $\omega : \mathcal{M}_n \to \mathcal{A}_\omega$. We now show that each of these mappings are one to one and onto, and finally that one is the inverse of the other.

Lemma 1 *The mapping $\omega : \mathcal{M}_n \to \mathcal{A}_\omega$ is one to one.*

Proof: We are required to show that $\omega(\mathcal{G}_1) = \omega(\mathcal{G}_2) \implies \mathcal{G}_1 = \mathcal{G}_2$. The proof is by contradiction.

Suppose $\mathcal{G}_1 \neq \mathcal{G}_2$ and $\omega(\mathcal{G}_1) = \omega(\mathcal{G}_2) = G_\omega$, say, with maximum weight $r > 0$ (that is, each of $\mathcal{G}_1$ and $\mathcal{G}_2$ contain at least one graph). Each of $\mathcal{G}_1$ and $\mathcal{G}_2$ is identical to an indexed collection of $n \times n$ adjacency matrices, say $[A_{(1)}, A_{(2)}, \ldots, A_{(n_1)}]$ and $[B_{(1)}, B_{(2)}, \ldots, B_{(n_2)}]$ respectively.

Let $a_{ij}^{[k]}$ denote the (i,j)th entry in $A_{(k)}$ and $b_{ij}^{[k]}$ denote the (i,j)th entry in $B_{(k)}$. Now $\omega(\mathcal{G}_1) = \omega(\mathcal{G}_2) = A_\omega$ implies

$$\sum_{k=1}^{n_1} a_{ij}^{[k]} = \sum_{k=1}^{n_2} b_{ij}^{[k]} = w_{ij}.$$

And the monotonicity of $\mathcal{G}_1$ and $\mathcal{G}_2$ implies that for all $l \leq \min(n_1, n_2)$

$$\sum_{k=1}^{l} a_{ij}^{[k]} = \sum_{k=1}^{l} b_{ij}^{[k]}.$$

If $n_1 = n_2$ then monotonicity implies $a_{ij}^{[k]} = b_{ij}^{[k]}$ for all k and $\mathcal{G}_1 = \mathcal{G}_2$, a contradiction. Otherwise, without loss of generality, suppose $n_1 > n_2$. Then for all $l > n_2$, we have $a_{ij}^{[l]} = 0$ for all (i,j). Since neither $\mathcal{G}_1$ nor $\mathcal{G}_2$ may contain empty graphs and be in $\mathcal{M}_n$, we must have $n_1 = n_2 = r$ and $\mathcal{G}_1 = \mathcal{G}_2$, a contradiction. □

Lemma 2 *The mapping $\omega : \mathcal{M}_n \to \mathcal{A}_\omega$ is onto. That is, $\forall A_\omega \in \mathcal{A}_\omega$, $\exists\, \mathcal{G} \in \mathcal{M}_n$ such that $A_\omega = \omega(\mathcal{G})$.*

Proof: For any $A_\omega \in \mathcal{A}_\omega$, we choose $\mathcal{G} = \gamma(A_\omega) \in \mathcal{M}_n$. To show that the mapping is onto, we now show that $A_\omega = \omega(\mathcal{G})$.

Let $\gamma(A_\omega) = [G_{(1)}, \ldots, G_{(r)}]$ with corresponding adjacency matrices $[A_{(1)}, \ldots, A_{(r)}]$ and denote the (i,j)th element of $A_{(k)}$ by $a_{ij}^{[k]}$. By definition, $a_{ij}^{[k]} = 1$ if and only if $k \leq w_{ij}$ and is zero otherwise. Consequently $\sum_{k=1}^{r} a_{ij}^{[k]} = w_{ij}$ and so $A_\omega = \sum_{k=1}^{r} A_{(k)}$, which implies $\omega(\gamma(A_\omega)) = \omega(\mathcal{G}) = A_\omega$. □

Lemma 3 *The mapping $\gamma : \mathcal{A}_\omega \to \mathcal{M}_n$ is one to one.*

Proof: We prove $\gamma(A_{\omega_1}) = \gamma(A_{\omega_2}) \implies A_{\omega_1} = A_{\omega_2}$ by proving its contrapositive: $A_{\omega_1} \neq A_{\omega_2} \implies \gamma(A_{\omega_1}) \neq \gamma(A_{\omega_2})$.

If we have two matrices $A_{\omega_1}, A_{\omega_2} \in \mathcal{A}_\omega$ such that $A_{\omega_1} \neq A_{\omega_2}$, then there exists at least one pair of indices i, j with $i \neq j$, such that the corresponding elements of A_{ω_1} and A_{ω_2} are $w_{ij}^{(1)}, w_{ij}^{(2)}$, respectively, and, without loss of generality, are such that $w_{ij}^{(1)} > w_{ij}^{(2)}$. Then, by definition of $\gamma(\cdot)$, the edge between i and j will appear in exactly $w_{ij}^{(1)} - w_{ij}^{(2)}$ many more graphs in $\gamma(A_{\omega_1})$ than in $\gamma(A_{\omega_2})$. Hence, $\gamma(A_{\omega_1}) \neq \gamma(A_{\omega_2})$. □

Lemma 4 *The mapping $\gamma : \mathcal{A}_\omega \to \mathcal{M}_n$ is onto. That is, $\forall \mathcal{G} \in \mathcal{M}_n$, $\exists\, A_\omega \in \mathcal{A}_\omega$, such that $\gamma(A_\omega) = \mathcal{G}$.*

Proof: For any $\mathcal{G} \in \mathcal{M}_n$, we choose $A_\omega = \omega(\mathcal{G}) \in \mathcal{A}_\omega$. Now consider $\gamma(A_\omega) \in \mathcal{M}_n$. As in the proof of Lemma 2, we may show that $\omega(\gamma(A_\omega)) = A_\omega$. Together, these two equalities imply $\omega(\gamma(A_\omega)) = \omega(\mathcal{G})$ and so, by Lemma 1, we have $\gamma(A_\omega) = \mathcal{G}$ as required. □

Theorem 1 *For $A \in \mathcal{A}_\omega$ and $\mathcal{G} \in \mathcal{M}_n$, $\omega(\mathcal{G}) = A \iff \mathcal{G} = \gamma(A)$; that is γ and ω are inverses on these domains.*

Proof: The proof follows directly from the preceding lemmas. □

The following results may be similarly proven.

1. The mapping $\omega : \mathcal{F} \to \mathcal{A}_\omega$ is many to one and onto.
2. The mapping $\omega : \mathcal{K}_n \to \mathcal{A}_\kappa$ is bijective with inverse mapping γ.
3. The mapping $\omega : \mathcal{C} \to \mathcal{A}_\rho$ is bijective with inverse mapping γ°.

References

Ashlock, D., Kim, E., and Guo, L. (2005), "Multi-clustering: Avoiding the Natural Shape of Underlying Metrics," *Smart Engineering System Design: Neural Networks, Evolutionary Programming, and Artificial Life*, 15, 453–461.

Bivand, R., Leisch, F., and Maechler, M. (2009), *pixmap: Bitmap Images ("Pixel Maps")*, r package version 0.4-10.

Carroll, J. D. and Corter, J. E. (1995), "A Graph-Theoretic Method for Organizing Overlapping Clusters into Trees, Multiple Trees, or Extended Trees," *Journal of Classification*, 12, 283–313.

Dimitriadou, E., Weingessel, A., and Hornik, K. (2001), "Voting-Merging: An Ensemble Method for Clustering," *Lecture Notes in Computer Science*, 2130, 217 – 224.

Donath, W. and Hoffman, A. (1973), "Lower bounds for the partitioning of graphs," *IBM J. Res. Develop.*, 17.

Ester, M., Kriegel, H., Sander, J., and Xu, X. (1996), "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proc. of KDD-96*.

Fraley, C. and Raftery, A. E. (1999), "MCLUST: Software for model-based cluster analysis," *Journal of Classification*, 16, 297–306.

Fred, A. (2001), "Finding consistent clusters in data partitions," in *Multiple Classifier Systems*, eds. Kittler, J. and Roli, F., Springer, vol. 2096 of *Lecture Notes in Computer Science*, pp. 309–318.

Fred, A. and Jain, A. K. (2002), "Evidence Accumulation Clustering Based on the K-Means Algorithm," in *Structural, Syntactic and Statistical Pattern Recognition*, pp. 442–451.

Godehardt, E. (1990), *Graphs as Structural Models*, vol. 4 of *Advances in Systems Analysis*, Braunschweig, Germany: Vieweg.

Hartigan, J. (1975), *Clustering Algorithms*, New York: John Wiley & Sons, Inc.

— (1985), "Statistical Theory in Clustering," *Journal of Classification*, 2, 63–76.

Hartigan, J. A. (1981), "Consistency of Single Linkage for High-Density Clusters," *Journal of the American Statistical Association*, 76, pp. 388–394.

MacQueen, J. (1967), "Some Methods for Classification and Analysis of Multivariate Observation," *Proc. 5th Berkeley Symp. Math. Stat. Prob.*, 1, 281–297.

Meila, M. and Shi, J. (2001), "A Random Walks View of Spectral Segmentation," in *8th International Workshop on Artificial Intelligence and Statistics*.

Michael Hahsler, K. H. and Buchta, C. (2008), "Getting things in order: An introduction to the R package seriation," *Journal of Statistical Software*, 25.

Shepard, R. and Arabie, P. (1975), "Additive cluster analysis of similarity data," in *Proceedings of the U.S.-Japan Seminar on Theory, Methods, and Applications of Multidimensional Scaling and Related Techniques. San Diego*.

— (1979), "Additive Clustering: Representation of Similarities as Combinations of Discrete Overlapping Properties," *Psychological Review*, 86, 87 – 123.

Sneath, P. H. and Sokal, R. R. (1973), *Numerical Taxonomy*, The Principles and Practice of Numerical Classification, San Francisco: W.H. Freeman and Company.

Strehl, A. and Ghosh, J. (2002), "Cluster Ensembles - A Knowledge Reuse Framework for Combining Multiple Partitions," *Journal of Machine Learning Research*, 3, 583 – 617.

Stuetzle, W. (2003), "Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample," *Journal of Classification*, 20, 25–47.

Topchy, A., Minaei-Bidgoli, B., Jain, A. K., and Punch, W. F. (2004), "Adaptive Clustering Ensembles," in *Proceedings of the 17th International Conference on Pattern Recognition*, pp. 272 – 275.

Tversky, A. (1977), "Features of Similarity," *Psychological Review*, 84, 327 – 352.

Zhou, W. (2010), "A New Framework for Clustering," Ph.D. thesis, University of Waterloo, Waterloo, Ontario, Canada.