1

Unsupervised Learning and Clustering

Pádraig Cunningham[1]

University College Dublin Padraig.Cunningham@ucd.ie

1.1 Introduction

P. CUNNINGHAM, H. BISCHOF, D. GREENE

The most fundamental distinction in Machine Learning is that between supervised and unsupervised techniques. Supervision invokes the idea of a teacher who guides the learning process. Typically this guidance comes in the form of labeled training examples that can be used to build a classification model. This external guidance is absent in unsupervised learning; thus the process of building a model from the data is more difficult. Often all that can be done is to cluster or organise the data in some way.

Unsupervised learning is very important in the processing of multimedia content, as clustering or partitioning of data in the absence of class labels is often a requirement. This chapter will present an overview of classic clustering techniques (k-Means, EM and Hierarchical) and will introduce the modern clustering techniques such as Kernel k-Means and Spectral Clustering. Given the popularity of SOMs in the processing of multimedia content, this topic will also be covered in detail. Because of the unsupervised nature of clustering, the validation of the resulting partition is a key issue: a comprehensive overview of cluster validation techniques will also be presented. This chapter will conclude with a review of unsupervised dimension reduction techniques.

1.2 Basic Clustering Techniques

1.2.1 *k*-Means Clustering

Partitional clustering methods involve directly decomposing a dataset into a flat partition consisting of k disjoint clusters, denoted $\mathcal{C} = \{C_1, \ldots, C_k\}$. These methods generally seek to produce a local approximation to a global objective function, which is identified by iteratively refining an initial solution.

1. Create an arbitrary initial clustering with centroids $\{\mu_1, \ldots, \mu_k\}$.
2. For each object $x_i \in \mathcal{X}$:
   1. Compute $||x_i - \mu_c||$ for $1 \leq c \leq k$.
   2. Reassign x_i to the cluster corresponding to the nearest centroid.
3. Update cluster centroids.
4. Repeat from Step 2 until a termination criterion is satisfied.

Fig. 1.1. Standard batch k-means algorithm.

Standard k-means is the most widely used partitional clustering algorithm. It employs an iterative relocation scheme to produce a k-way hard clustering that locally minimises the distortion between the data objects and a set of k cluster representatives. Each representative, referred to as a *centroid*, is computed as the mean vector of all objects assigned to a given cluster. In the classical version of the algorithm, distortion is measured using Euclidean distance, so that the goal of the clustering process becomes the minimisation of the *sum-of-squared error* (SSE) between the objects and cluster centroids $\{\mu_1, \ldots, \mu_k\}$:

$$SSE(\mathcal{C}) = \sum_{c=1}^{k} \sum_{x_i \in C_c} ||x_i - \mu_c||^2 \quad \text{where} \quad \mu_c = \frac{\sum_{x_i \in C_c} x_i}{|C_c|} \tag{1.1}$$

While many variations of the basic algorithm exist, the most frequently applied version for offline clustering is the *batch k-means* algorithm, generally attributed to Forgy [21], which involves a two-step process as shown in Figure 1.1. In the first step, each object is reassigned to the closest cluster centroid. Once all objects have been processed, the centroid vectors are updated to reflect the new cluster assignments. The iterative refinement process is repeated until a given termination criterion is satisfied. Typically this occurs when the assignment of objects to clusters no longer changes from one iteration to another. Alternatively, the procedure may be terminated if the change in the evaluation of Eqn. 1.1 between two successive iterations is less than a user-defined threshold.

The SSE function (1.1) implicitly assumes that the clusters approximate a mixture of Gaussians, such that each cluster is spherical in shape and data objects are largely concentrated near its centroid. Consequently, k-means will often fail to identify a useful partition in cases where the clusters are non-spherical or differ significantly in size. The traditional objective for k-means can also give undue influence to outlying objects. Their effect in centroid construction can lead to vectors that are not representative of the underlying groups in the data, resulting in highly skewed clusters. Some authors have proposed the introduction of an "outlier cluster", which is used to hold objects that do not fit well in any other cluster [20]. Others have suggested repeatedly applying the clustering algorithm and removing poorly clustered

data after each run [27]. However, both approaches require the introduction of an arbitrary threshold to determine whether an object is far enough from its current centroid to be deemed an outlier. Another problem occurs when the iterative refinement process results in the formation of empty clusters. A common strategy to deal with this is to assign the most outlying object (*i.e.* furthest from its current centroid) to the empty cluster. However, if the problem persists, it is more likely that the fault may lie with the choice of clustering model, such as the use of an unsuitable value for k.

1.2.2 Fuzzy Clustering

Dunn [17] proposed a generalisation of standard k-means, the Fuzzy c-means (FCM) algorithm, which allows objects to belong to different clusters to certain degrees as expressed by probabilistic weights. These weights may be represented in the form of a $n \times k$ matrix $\mathbf{V}$, where $V_{ij} \in [0, 1]$ denotes the degree of membership of the object x_i in cluster C_j, and $\sum_j V_{ij} = 1$. Once again, the task of clustering is to minimise the distortion between objects and centroids, which is now measured by the fuzzy criterion function

$$F(\mathcal{C}, \mathbf{V}) = \sum_{i=1}^{n} \sum_{j=1}^{k} {V_{ij}}^m \, ||x_i - \mu_j||^2 \tag{1.2}$$

where the exponent $m > 1$ controls the fuzziness of object memberships. In this algorithm, centroids are computed using:

$$\mu_j = \frac{\sum_{i=1}^{n} {V_{ij}}^m x_i}{\sum_{i=1}^{n} {V_{ij}}^m} \tag{1.3}$$

Another well-known soft partitional clustering technique is the Expectation Maximisation (EM) algorithm [10]. Unlike the other techniques described here, this algorithm takes a model-based approach to identifying groups in data. Formally, EM clustering is based on the assumption that the data objects are generated using a model θ which consists of a mixture of k underlying probability distributions $\{\theta_1, \ldots, \theta_k\}$. The task of clustering can then be viewed as the problem of determining the most likely parameters for the model, where each component in the mixture represents a cluster. The likelihood of an object x_i is given by:

$$P(x_i|\theta) = \sum_{c=1}^{k} P(C_c) P(x_i|C_c)$$

In the standard formulation of the algorithm, the k distributions are assumed to be Gaussians, so that the problem becomes the approximation of the mean and covariance of each component. In practice, the algorithm begins with an initial estimate for the model parameters and subsequently applies an iterative optimisation approach that alternates between two steps: firstly identify

the expected value of the log likelihood with respect to the current parameter estimates, then find new parameter values to maximise this likelihood. Once the algorithm has converged to a local solution, each data object is probabilistically assigned to each cluster based on the estimated distributions. As with standard k-means, the choice of initial clusters can have a considerable effect on the accuracy of the final solution.

1.2.3 Hierarchical Clustering

Instead of generating a flat partition of data, it may often be useful to construct a hierarchy of concepts by producing a set of nested clusters that may be arranged to form a tree structure. While partitional clustering methods have received more attention in recent literature, hierarchical clustering algorithms represent the traditional choice for performing document clustering, since text collections often contain broad themes that may be naturally sub-divided into more specific topics. Hierarchical algorithms are generally organised into two distinct categories:

Agglomerative: Begin with each object assigned to a singleton cluster. Apply a bottom-up strategy where, at each step, the most similar pair of clusters are merged.

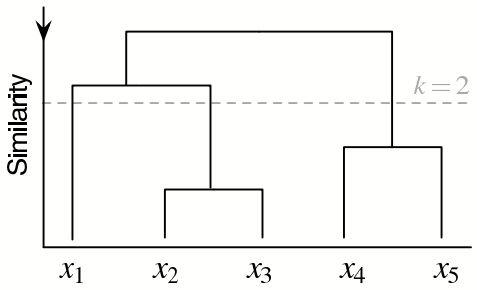
Divisive: Begin with a single cluster containing all n objects. Apply a top-down strategy where, at each step, a chosen cluster is split into two sub-clusters.

In either case, the resulting hierarchy may be presented visually using a tree-like structure referred to as a *dendrogram*, which contains nodes for each cluster constructed by the clustering algorithm, together with cluster relations illustrating the merge or split operations that were performed during the clustering process. Figure 1.2 provides a simple example of an agglomerative clustering process applied to a set of five data objects, together with the corresponding cluster assignments. It is worth noting that, as each merge operation is performed, the similarity between the chosen pair of cluster decreases.

Unlike the requirement in most partitional algorithms to specify a value for the number of clusters k in advance, hierarchical algorithms support the construction of a tree from which a user may manually select k by examining the resulting dendrogram and identifying an appropriate cut-off point [41]. For instance, by cutting the tree in Figure 1.2 at the level indicated, we can derive a clustering of the data for $k = 2$ from the two leaf nodes at that level.

Agglomerative Algorithms

Agglomerative hierarchical clustering (AHC) involves the construction of a tree of clusters from the bottom upwards. A variety of agglomerative algorithms have been proposed, such as BIRCH [55] and CURE [25], which are



$\mathcal{C} = \{\{x_1, x_2, x_3, x_4, x_5\}\}$

$\mathcal{C} = \{\{x_1, x_2, x_3\}, \{x_4, x_5\}\}$

$\mathcal{C} = \{\{x_1\}, \{x_2, x_3\}, \{x_4, x_5\}\}$

$\mathcal{C} = \{\{x_1\}, \{x_2, x_3\}, \{x_4\}, \{x_5\}\}$

$\mathcal{C} = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}\}$

Fig. 1.2. Example dendrogram representing an agglomerative clustering of five data objects, together with the corresponding cluster memberships.

suitable for specific types of data. However, we focus on the standard formulation that has widely been used in a range of domains, which proceeds as follows:

1. Assign each object to a singleton clusters.
2. Update the pairwise inter-cluster similarity matrix.
3. Identify and merge the most similar pair of clusters.
4. Repeat from Step 2 until a single cluster remains or a given termination criterion has been satisfied.

When an estimation for the number of clusters k is given in advance, the algorithm may be terminated when the required number of leaf nodes remain in the dendrogram.

A variety of *linkage* strategies exist for determining which pair of clusters should be merged from among all possible pairs. While these strategies are typically expressed in terms of distances, they may be easily adapted to use similarity values such as those produced by the cosine measure. Given a symmetric matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$, where S_{ij} denotes the similarity between a pair of objects x_i and x_j, the most popular linkage strategies are defined as follows:

Single linkage: The most common strategy, also known as the nearest neighbour technique, defines the similarity between two clusters (C_a, C_b) as the maximum similarity between an object assigned to C_a and an object assigned to C_b:

$$sim(C_a, C_b) = \max_{x_i \in C_a, x_j \in C_b} S_{ij}$$

While this approach is widely used, it can often produce clusters of poor quality as it is subject to the phenomenon of "chaining", where singletons are repeatedly merged with an existing cluster, resulting in one large, elongated cluster with highly dissimilar objects at either end.

Complete linkage: The similarity between two clusters (C_a, C_b) is defined as the minimum similarity between an object assigned to C_a and an object assigned to C_b:

$$sim(C_a, C_b) = \min_{x_i \in C_a, x_j \in C_b} S_{ij}$$

This strategy tends to favour strongly compact, tightly coupled clusters and is often highly sensitive to the presence of outliers.

Average linkage: The similarity between a pair of clusters (C_a, C_b) is calculated as the mean similarity between objects assigned to C_a and objects assigned to C_b:

$$sim(C_a, C_b) = \frac{\sum_{x_i \in C_a} \sum_{x_j \in C_b} S_{ij}}{|C_a|\,|C_b|}$$

This strategy is often referred to as *unweighted pair group method using arithmetic averages* (UPGMA), since normalising by cluster size has the effect of giving equal weights to objects that are assigned to clusters of different sizes.

Clearly, the choice of linkage strategy can significantly affect the structure of the clusters that are generated by AHC. As a consequence, the prior selection of a suitable strategy for a given dataset may represent a non-trivial parameter selection problem. In practice, a user may generate several hierarchies using different approaches, and manually inspect the results to choose the most appropriate solution.

Limitations of Agglomerative Algorithms

A substantial drawback of standard agglomerative algorithms is that poor decisions made early in the clustering process can greatly influence the accuracy of the final solution. Without the use of a global objective function, many potential mergers at these stages may appear to be equally valid. Once a merging decision has been made, there exists no facility to rectify an erroneous choice at a later stage. On the contrary, the adverse effects of these decisions are often exaggerated as the clustering process continues. In addition to deficiencies in clustering accuracy, hierarchical clustering algorithms are generally considerably more computationally costly than their partitional counterparts, typically having time complexity $O(n^3)$.

Divisive Algorithms

In contrast to agglomerative methods, divisive hierarchical clustering involves building a cluster tree from the root node downwards.

1. Assign all data objects to a single cluster.
2. Select a cluster to split.
3. Replace the selected cluster with two new sub-clusters.
4. Repeat from Step 2 until k leaf clusters have been generated or a given termination criterion has been satisfied.

1. Assign all n objects to a single cluster.
2. Select a cluster C_c to split according to a chosen splitting criterion.
3. Generate τ 2-way partitions of the cluster C_c using randomly initialised k-means.
4. Replace C_c with the best pair of clusters as determined by a given clustering criterion.
5. Repeat from Step 2 until k leaf clusters have been generated.

Fig. 1.3. Bisecting k-means (BKM) algorithm.

Several authors have empirically shown divisive algorithms to be superior to agglomerative techniques on text data [13]. In addition, these algorithms are often less time consuming than traditional bottom-up clustering. However, in general, they have been employed less frequently due to the non-trivial problems of selecting a cluster to split and finding the optimal sub-division of the chosen cluster.

Bisecting k-means

As a representative example of divisive clustering, we consider the algorithm proposed by Steinbach *et al.* [50], which combines aspects of hierarchical and partitional clustering. Initially, all objects are assigned to a single root cluster. The algorithm involves repeatedly selecting an existing cluster and splitting the cluster into two sub-clustering using the generalised k-means algorithm with cosine similarity. The process is repeated until k clusters have been obtained. To split a cluster, a fixed number of randomly-initialised bisections τ may be performed, from which the best candidate is selected. This choice is determined by a cluster evaluation criterion, such as mean document-centroid distance:

$$Cen(C_c) = \frac{\sum_{x_i \in C_c} d(x_i, \mu_c)}{|C_c|} \tag{1.4}$$

A larger value for τ renders the algorithm less sensitive to the choice of initial clusters than the partitional algorithms described previously, although it does increase the computational cost of applying the algorithm. A summary of the complete procedure is given in Figure 1.3.

Several strategies have been proposed to identify the most appropriate cluster to split. A naïve approach is to divide the largest cluster into two sub-clusters at each stage [50]. However, this may be inappropriate when working with text corpora, which frequently contain "unbalanced" clusters that differ in their relative proportions. An alternative strategy is to split the cluster with maximal distortion [13]. Given k potential candidates for splitting, Zhao & Karypis [57] proposed evaluating all possible candidates and selecting the split that leads to the best subsequent partition containing $k + 1$ clusters.

However, this approach requires significantly more computational time than the standard formulation of the algorithm.

1.3 Modern Clustering Techniques

1.3.1 Kernel Clustering

Kernel methods involve the transformation of a dataset to a new, possibly high-dimensional space where non-linear relationships between objects may be more easily identified. Rather than explicitly computing the transformed representation $\phi(x)$ of each data object x, the application of the "kernel trick" [1] allows us to consider the affinity between a pair of objects (x_i,x_j) using a kernel function κ, which is defined in terms of the dot product:

$$\kappa(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \tag{1.5}$$

In practice, the function κ is represented by an $n \times n$ symmetric, positive semi-definite *kernel matrix* (or Gram matrix) $\mathbf{K}$, such that $K_{ij} = \kappa(x_i, x_j)$. By re-formulating algorithms using only dot products and subsequently replacing these with affinity values from $\mathbf{K}$, we can efficiently apply learning algorithms in the new kernel space.

Another significant advantage of kernel methods is their modularity, where every method is composed of two decoupled components: a generic learning algorithm, and a problem-specific kernel function. Consequently, it is possible to develop algorithms that can readily be deployed in a wide range of domains without requiring any customisation. Novel kernels can also be constructed in a modular fashion by chaining together multiple existing functions together.

The main focus of research in this area has been on the development of techniques for supervised tasks, notably the well-known *support vector machine* (SVM) classifier [8]. However, kernel methods have also been shown to be effective in unsupervised problems [12]. We now describe several algorithms and kernel functions that are relevant in this context.

Kernel *k*-means

A variety of popular clustering techniques have been re-formulated for use in a kernel-induced space, including the standard k-means algorithm. To describe the algorithm, we firstly observe that, using the notation given above, the squared Euclidean distance between a pair of objects in the kernel space represented by a matrix $\mathbf{K}$ can be expressed as:

$$||\phi(x_i) - \phi(x_j)||^2 = K_{ii} + K_{jj} - 2K_{ij} \tag{1.6}$$

This may be used as a starting point for the identification of cluster structures. Formally, given a set of objects $\{x_1, \ldots, x_n\}$, the kernel k-means algorithm

1. Select k arbitrary initial clusters $\{C_1, \ldots, C_k\}$.
2. For each object $x_i \in \mathcal{X}$ and centroid μ_c, compute the distance:

$$d(x_i, \mu_c) = K_{ii} + \frac{\sum_{x_j, x_l \in C_c} K_{jl}}{|C_c|^2} - \frac{2 \sum_{x_j \in C_c} K_{ij}}{|C_c|}$$

3. Assign each x_i to cluster corresponding to nearest centroid.
4. Repeat from Step 2 until termination criterion is satisfied.

Fig. 1.4. Kernel k-means (KKM) algorithm.

[47] seeks to minimise the distortion between the objects and the "pseudo-centroids" $\{\mu_1, \ldots, \mu_k\}$ in the new space:

$$\sum_{c=1}^{k} \sum_{x_i \in C_c} ||\phi(x_i) - \mu_c||^2 \quad \text{where} \quad \mu_c = \frac{\sum_{x_i \in C_c} \phi(x_i)}{|C_c|} \tag{1.7}$$

Note that this expression is analogous to the SSE objective (1.1) used in standard k-means. However, rather than explicitly constructing centroid vectors in the kernel space, distances are computed using dot products only. From Eqn. 1.6, we can formulate squared object-centroid distance by the expression:

$$||\phi(x_i) - \mu_c||^2 = K_{ii} + \frac{\sum_{x_j, x_l \in C_c} K_{jl}}{|C_c|^2} - \frac{2 \sum_{x_j \in C_c} K_{ij}}{|C_c|} \tag{1.8}$$

The first term above may be excluded as it remains constant; the second is a common term representing the self-similarity of the centroid μ_c, which only needs to be calculated once for each cluster; the third term represents the affinity between x_i and the centroid of the cluster C_c.

This kernelised algorithm has a significant advantage over standard k-means in the sense that, given an appropriate kernel function, it can be used to identify structures that are not necessarily spherical or convex. In addition, once we have constructed a single matrix $\mathbf{K}$, multiple partitions may be subsequently generated without referring back to the original feature space.

1.3.2 Spectral Clustering

Motivated by work in graph theory, unsupervised feature extraction methods have been developed that employ well-known techniques from linear algebra to analyse the spectral properties of a graph representing a dataset. In practice, this involves constructing a reduced dimensional space from the eigenvalue decomposition (EVD) of a matrix form of the graph. Existing clustering algorithms may subsequently be applied in the reduced space to uncover the underlying classes in the data. Spectral clustering methods have been widely

used due to their efficiency and applicability in a variety of tasks, including image segmentation [49], gene expression analysis [34] and document clustering [11].

Graph Partitioning

A common way of expressing the relations between pairs of data objects is to use a symmetric similarity or affinity matrix $\mathbf{S}$, where S_{ij} denotes the association between the objects x_i and x_j. The task of producing a disjoint clustering may then be modelled as a graph partitioning problem, where $\mathbf{S}$ becomes the adjacency matrix for a weighted undirected graph $G(\mathcal{V}, \mathcal{E})$. In this model, the set of vertices $\mathcal{V}$ represents the data objects and the set of edges $\mathcal{E}$ represents pairwise similarities between objects. Using this graph-theoretic formulation, clustering becomes the problem of finding the partition $\{V_1, \ldots, V_k\}$ of G that optimises a given cost criterion.

A variety of criteria have been used in graph partitioning, which are also relevant to the theoretical foundations of spectral clustering. The simplest of these, the *minimum cut* criterion, measures the weight of the edges crossing the partition. Formally, the optimisation of the criterion involves locating a bi-partition (C_1, C_2) of the graph vertices such that $C_1 \cup C_2 = \mathcal{V}$, which minimises the sum of the weights of edges connecting the two clusters, as denoted by:

$$s(C_1, C_2) = \sum_{i \in C_1, j \in C_2} S_{ij} \tag{1.9}$$

This expression shows that the weight of the cut is directly proportional to the number of edges that join the two sub-graphs. Consequently, the criterion favours small groups of isolated vertices. This makes it sensitive to outliers and often leads to highly unbalanced clusterings.

A more robust measure for assessing bi-partitions, the *normalised cut* criterion, was proposed in [48]. This measures the degree of association between a cluster and the remaining vertices, relative to the total association within that cluster:

$$Ncut(C_1, C_2) = \frac{s(C_1, C_2)}{s(C_1, \mathcal{V})} + \frac{s(C_1, C_2)}{s(C_2, \mathcal{V})} \tag{1.10}$$

The normalisation given in the denominator makes the criterion less sensitive to the presence of outlying objects. Yu & Shi [54] subsequently generalised this objective for multi-class partitioning:

$$KNcut(\mathcal{C}) = \sum_{i=1}^{k} \frac{s(C_i, \mathcal{V} \backslash C_i)}{s(C_i, \mathcal{V})} \tag{1.11}$$

Spectral Bi-partitioning

Unfortunately, the problem of finding an optimal partition according to the criteria described in the previous section is NP-complete. While traditional

techniques such as the Kernighan-Lin algorithm [33] have been used to produce local approximations, such methods often have drawbacks in terms of partition accuracy. Rather than directly attempting to optimise a given criterion, many authors have sought to transform the optimisation task into a generalised eigenvalue problem. Given a symmetric adjacency matrix $\mathbf{S}$, this involves computing its eigenvalue decomposition:

$$\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathsf{T}}$$

where the diagonal entries of $\mathbf{\Lambda}$ represent the set of eigenvalues and the columns of $\mathbf{V}$ are a corresponding set of orthogonal eigenvectors. Unlike local partitioning methods, analysing the spectrum of $\mathbf{S}$ allows grouping to be performed based on global information describing the structure of the corresponding graph.

Early work in this area [14, 19] indicated the existence of a connection between the problem of finding vertex separators for a graph and the eigenvalue decomposition of its corresponding *Laplacian matrix* $\mathbf{L} = \mathbf{D} - \mathbf{S}$, where $\mathbf{D}$ denotes a diagonal degree matrix such that $D_{ii} = \sum_{j=1}^{n} S_{ij}$. The Laplacian of a graph G containing n vertices is symmetric positive semi-definite, with non-negative eigenvalues $0 = \lambda_1 < \lambda_2 < \cdots \leq \lambda_n$ and corresponding eigenvectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$. The most important common observation made by these authors was that the spectrum of a graph provides useful structural information that may indicate how best to partition its vertices. The use of spectral partitioning was popularised by the proposal of a formal technique by Pothen *et al.* [43]. Following from the discussion given in [26], a bi-partition (C_1, C_2) of G may be represented by a membership indicator vector $\mathbf{q} = \{q_1, \ldots, q_n\}$ such that:

$$q_i = \begin{cases} +1 & \text{if } i \in C_1 \\ -1 & \text{if } i \in C_2 \end{cases}$$

If the adjacency matrix $\mathbf{S}$ has a block-diagonal structure (*i.e.* the rows can be reorganised by cluster membership to form a checker-board pattern), we can optimise the minimum cut by finding a clustering that minimises the sum of the weights in the off-diagonal blocks. The problem can be formulated as the search for a vector $\mathbf{q}$ that minimises:

$$\underset{q}{\operatorname{argmin}} \sum_{i,j=1}^{n} S_{ij}(q_i - q_j)^2 \quad \text{such that} \quad \sum_{i=1} q_i^2 = 1$$

This objective can also be expressed in quadratic form using the Laplacian $\mathbf{L}$:

$$\underset{q}{\operatorname{argmin}} \; q\mathbf{L}\mathbf{q}$$

Rather than solving this as a complex combinatorial problem, a solution may be found by relaxing the requirement on $\mathbf{q}$ to contain discrete values, so that the assignment of each vertex is continuous, with membership weights taking

real values in the range $[-1, 1]$. The partition approximating the minimal cut can then be found by examining the eigenvectors of $\mathbf{L}$. Specifically, the relaxed membership weights are calculated as the components of the eigenvector $\mathbf{v}_2$ corresponding to the second smallest eigenvalue λ_2 of the Laplacian (*i.e.* the first non-trivial eigenvector), which is often referred to as the *Fiedler vector*.

A variety of justifications for partitioning based on $\mathbf{v}_2$ are given in the literature. Fiedler [19] showed the association between its corresponding eigenvalue λ_2 and the edge connectivity of a graph, while Pothen *et al.* [43] demonstrated a relationship between the edge separator induced by $\mathbf{v}_2$ and the *isoperimetric number* of a graph, which represents the value of the smallest possible edge cut over all candidate separators. The latter has motivated several popular spectral bi-partitioning algorithms. In these, the vertices are sorted according to the values in $\mathbf{v}_2$, and those vertices with values below a chosen threshold, such as 0, the mean value or the median value, are assigned to one cluster, with the remaining vertices assigned to the second cluster.

Spectral methods have also been developed to optimise other, more robust graph partitioning criteria. Notably, in [48] a spectral approach was proposed to find a bisection that minimises the normalised cut criterion (1.10). A good approximation may be identified in a manner similar to that described previously, but rather using the spectrum of the *normalised Laplacian matrix* of the graph, which is defined as $\mathbf{L_n} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{S})\mathbf{D}^{-\frac{1}{2}}$. Two clusters are formed from the normalised Fiedler vector by sorting the entries and choosing a splitting point along the vector which results in the minimal value for Eqn. 1.10.

K-Way Spectral Clustering

In most cases, we will typically want to partition a dataset into more than two clusters. Two general approaches have been proposed in the literature to extend spectral bi-partitioning to the problem of k-way clustering. The first involves recursively applying spectral bi-partitioning to hierarchically divide each resulting sub-graph until k clusters have been recovered [48]. However, if k is not a power of 2, it is unclear as to how to choose which segments should be sub-divided.

A more effective approach involves directly producing a k-way partition by constructing an embedding from multiple eigenvectors of the affinity matrix. However, rather than using those vectors corresponding to the smallest eigenvalues, clustering may be performed using the eigenvectors associated with the largest eigenvalues, which also contain structural information. A formal justification for the benefits of clustering in the reduced space formed from these vectors was given in the *polarisation theorem* proposed in [6]. This theorem asserts that, as an affinity matrix $\mathbf{S}$ is projected onto smaller subsets of successive leading eigenvectors, the angles between highly similar objects are least distorted, while the angles between dissimilar objects tend to greatly increase. Consequently, by magnifying the similarities between objects that belong to

the same natural class and attenuating the associations between objects belonging to different classes, the clustering problem will often become easier to solve.

K-way spectral clustering techniques generally consist of three principal phases: preprocessing, spectral mapping and post-processing [53]. We now describe each of these phases individually and summarise the most popular approaches that have been used to implement them.

Preprocessing: Initially, an affinity matrix $\mathbf{S}$ is constructed from the original data using an appropriate metric, such as the Gaussian kernel function for image data or cosine similarity for text documents. As with bipartitioning, various normalisation techniques may be applied to $\mathbf{S}$ to support the optimisation of different partitioning criteria. Most commonly, an approximation to the k-way normalised cut (1.11) has been used, which is found by computing the truncated EVD of the normalised affinity matrix given by:

$$\mathbf{S_n} = \mathbf{D}^{-\frac{1}{2}}\mathbf{S}\mathbf{D}^{-\frac{1}{2}} \tag{1.12}$$

When using this objective, some authors have observed that removing the influence of the diagonal values by setting $S_{ii} = 0$ prior to decomposition results in improved accuracy [42].

Spectral mapping: The second phase of the spectral clustering process involves computing the eigenvalue decomposition of the normalised affinity matrix. In [42] it was shown that, when partitioning data into k clusters, the use of eigenvectors corresponding to the k largest eigenvalues affords the best discriminating power. By stacking these vectors in columns to form $\mathbf{Y} \in \mathbb{R}^{n \times k}$, a reduced-dimensional representation is produced for the original n objects.

Post-processing: The columns of a k-dimensional spectral embedding $\mathbf{Y}$ can be viewed as a set of k semantic variables. However, since these variables may take negative values, they are not immediately interpretable as clusters. In simple cases where the affinity matrix is approximately block diagonal, it may be possible to identify a partition by inspecting the values in $\mathbf{Y}$. However, for real-world data such as text corpora some form of post-processing will be required to extract the final cluster assignments. A popular approach is to treat the rows $\{y_1, \ldots, y_k\}$ as points in a geometric space $\mathbb{R}^k$ and apply a partitional algorithm, such as standard k-means, to cluster these points. A final clustering of the original dataset may be derived by simply assigning the object x_i to the cluster C_j which contains the corresponding embedded point y_j. It has been shown the quality of this partition may often be improved by normalising the rows of $\mathbf{Y}$ to L2 unit length prior to clustering [42]. Several other authors have focused on directly decomposing the selected eigenvectors into a set of k clusters without the need for the subsequent application of a clustering algorithm [54].

1. Construct an affinity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$ on the original data $\mathcal{X}$, and set $S_{ii} = 0$.
2. Form the normalised affinity matrix:

$$\mathbf{S_n} = \mathbf{D}^{-\frac{1}{2}}\mathbf{S}\mathbf{D}^{-\frac{1}{2}}$$

3. Decompose $\mathbf{S_n}$ and construct an embedding $\mathbf{Y} \in \mathbb{R}^{n \times k}$, such that the columns are given by the eigenvectors corresponding to the k largest eigenvalues.
4. Normalise the rows of $\mathbf{Y}$ to L2 unit length.
5. Apply standard k-means to the rows of $\mathbf{Y}$ to generate a k-way clustering $\mathcal{C}$.
6. Produce a clustering of $\mathcal{X}$ by assigning each object x_i to the j-th cluster if $y_i \in C_j$.

Fig. 1.5. Ng-Jordan-Weiss (NJW) spectral clustering algorithm.

To illustrate how the three phases fit together, a summary of a popular representative algorithm, Ng-Jordan-Weiss (NJW) clustering [42], is given in Figure 1.5.

Bipartite Spectral Co-clustering

The techniques described previously in this section focus solely on the problem of grouping the objects in a dataset. However, in certain situations it may be useful to perform *co-clustering*, where both objects and features are assigned to groups simultaneously. Such techniques are related to the *principle of the duality of clustering objects and features*, which states that a clustering of objects induces a clustering of features while a clustering of features also induces a clustering of objects [11]. The co-clustering problem may be viewed as the task of partitioning a weighted bipartite graph. Formally, we build a graph $G(\mathcal{V}, \mathcal{E})$ such that $\mathcal{V} = \mathcal{V}_X \cup \mathcal{V}_T$, where $\mathcal{V}_X$ is a set of vertices representing the n objects and $\mathcal{V}_T$ is a set of vertices representing the m features. Feature values are given by the weights on the edges (i, j) in $\mathcal{E}$. We can conveniently represent such a graph using a feature-object matrix $\mathbf{A}$.

While the methods in the previous section involve analysing the eigendecomposition of an affinity matrix, for the bipartite case several authors have suggested the use of the related *singular value decomposition* (SVD), which may be applied to rectangular matrices. Formally, this involves decomposing a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ into the product of three factors:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}} \tag{1.13}$$

The columns of the matrix $\mathbf{U} \in \mathbb{R}^{m \times m}$ are referred to as the left singular vectors, the rows of $\mathbf{V} \in \mathbb{R}^{n \times n}$ are the right singular vectors, and the diagonal entries of $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ are called the singular values of $\mathbf{A}$. Note that the left

singular vectors are equivalent to the eigenvectors of $\mathbf{A}\mathbf{A}^{\mathsf{T}}$, the right singular vectors are the eigenvectors of $\mathbf{A}^{\mathsf{T}}\mathbf{A}$ and their identical sets of eigenvalues are given by the diagonal of $\mathbf{\Sigma}^2$.

Dhillon [11] suggested that an approximation for the optimal normalised cut of a bipartite graph represented by a matrix $\mathbf{A}$ may be obtained by analysing the $l = \log_2 k$ leading singular vectors of the degree-normalised matrix given by:

$$\mathbf{A_n} = \mathbf{D_1}^{-\frac{1}{2}}\mathbf{A}\mathbf{D_2}^{-\frac{1}{2}} \tag{1.14}$$

where $\mathbf{D_1}$ and $\mathbf{D_2}$ are diagonal matrices such that

$$[D_1]_{ii} = \sum_{j=1}^{n} A_{ij}\,, \quad [D_2]_{jj} = \sum_{i=1}^{m} A_{ij} \tag{1.15}$$

If $\mathbf{A}$ is a feature-object matrix, the rows of the left truncated vectors $\mathbf{U_l}$ represent a l-dimensional embedding of the features, while the columns of the right truncated vectors $\mathbf{V_l}$ represent an embedding of the data objects. By selecting the leading vectors of the spectral decomposition, we can produce a reduced-dimensional space that amplifies the natural structures in the data. In this case, a unified embedding $\mathbf{Z} \in \mathbb{R}^{(m+n)\times l}$ is constructed by normalising and arranging the truncated factors as follows:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{D_1}^{-1/2}\mathbf{U_l} \\ \mathbf{D_2}^{-1/2}\mathbf{V_l} \end{bmatrix}$$

A partitional clustering algorithm, such as k-means, is then applied in the geometric space $\mathbf{Z}$ to produce a simultaneous k-way partitioning of both objects and features.

1.4 Self Organising Maps

1.5 Cluster Validation

We now consider the task of assessing the validity of the output of a clustering algorithm, which represents a fundamental problem in unsupervised learning. Unlike in classification tasks, cluster analysis procedures will generally be unable to refer to predefined class labels when employed in real-world applications. Consequently, there is no clear definition of what constitutes a correct clustering for a given dataset. As a result, it may be difficult to distinguish between a solution consisting of groups that accurately reflect the underlying patterns in the data and one that does not provide the user with any helpful insight. While it may be possible in some cases for a domain expert to manually evaluate a clustering solution, this will be unfeasible for larger datasets and may introduce an element of human bias.

In contrast, *cluster validation* methods automatically produce a quantitative evaluation, which can be highly useful both in the exploratory analysis of data and in the design of new clustering algorithms. The validation problem can be viewed as comprising of several different tasks:

Examining cluster tendency: In certain applications, a crucial initial step in the cluster analysis process is to determine whether any significant structures exist in a dataset at all. However, in the document clustering literature it has been common to assume that text corpora will contain at least two identifiable topics.

Model selection: This task relates to the identification of an appropriate clustering algorithm and a corresponding set of parameter values. In the context of document clustering, a particularly important model selection problem is that of estimating the optimal number of clusters in a corpus, denoted by $\hat{k}$. For certain datasets, there may be several reasonable values for $\hat{k}$.

Relative comparison: It is often necessary to directly compare two or more candidate clusterings of the same dataset. This comparison may be performed as part of model selection, or may be used to evaluate the performance of a newly proposed clustering algorithm, relative to that afforded by existing algorithms.

Stability analysis: When a clustering solution is generated using an algorithm that contains a stochastic element or requires the selection of key parameter values, it is important to consider whether the solution represents a "definitive" solution that may easily be replicated. This can typically be determined by assessing the level of pairwise agreement between two or more clusterings of the same data.

A wide variety of validation methods have been proposed in the cluster analysis literature, which pertain to one or more of the above tasks. In the remainder of this chapter we review a range of methods, both classical and contemporary, many of which are relevant for document clustering. These methods are often organised into three distinct categories:

1. *Internal validation:* Compare clustering solutions based on the goodness of fit between each clustering and the raw data on which the solutions were generated.
2. *External validation:* Assess the agreement between the output of a clustering algorithm and a predefined reference partition that is unavailable during the clustering process.
3. *Stability-based validation:* Evaluate the suitability of a given clustering model by examining the consistency of solutions generated by the model over multiple trials.

1.5.1 Internal Validation

Internal validation methods are designed to provide a means of systematically assessing the quality of a clustering based on some evaluation function, which usually takes the form of an index measuring the goodness of fit between the clustering and the data on which it was generated. This evaluation is based solely on aspects of the features and metrics used during the clustering process, without considering any additional information or external supervision. In certain cases, these indices can also be used to provide an objective function for clustering, although many are intractable to optimise directly. Consequently, internal validation techniques are generally applied after the completion of the clustering procedure.

Model selection in areas such as bioinformatics has frequently been performed by using internal techniques [5]. Specifically, it is common to generate multiple clusterings of the data for a range of reasonable parameter values. For knowledge discovery tasks, it may be necessary to repeatedly adjust parameter values and reapply the clustering algorithm until a useful solution is obtained. To guide this process, one or more internal validation indices may be employed to assess the quality of different solutions. A set of suitable parameter values may be identified by locating a solution which optimises these indices.

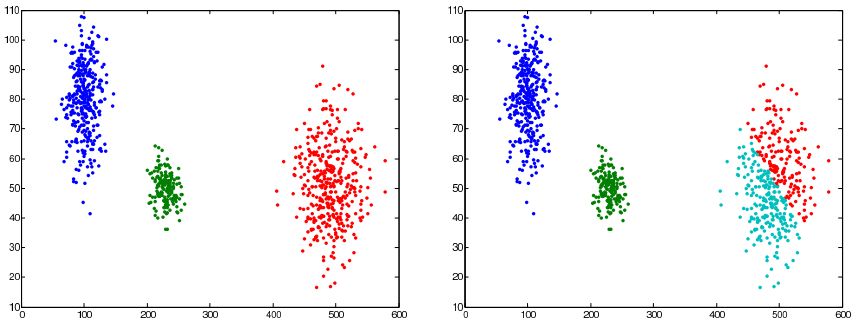
It is common for internal validation indices to measure goodness-of-fit by examining aspects of a clustering solution such as intra-cluster compactness and inter-cluster separation. To illustrate this idea, we consider the simple two-dimensional data depicted in Figure 1.6, for which we wish to choose a suitable value for the number of clusters k. An internal index is likely to favour the first clustering in Figure 6(a), which consists of three clusters that are relatively compact and well-separated. However, the partition shown in Figure 6(b) is clearly a poor fit for the data, as two of the clusters are not well-separated. Therefore, evaluating the second partition with the same index should result in a relatively poor score. From this, we can conclude that $k = 3$ is likely to represent a more suitable choice for the data.

We now discuss a number of internal indices that have been traditionally used to evaluate hard clusterings.

Calinski-Harabasz index

Motivated by the clustering objectives used in well-known partitional algorithms, a number of internal indices have been proposed which assess cluster quality by considering the squared distances between data objects and cluster representatives. Formally, the *within-cluster sum of squares* is the total of the squared distances between each object x_i and the centroid of the cluster C_c to which it has been assigned:

$$W(\mathcal{C}) = \sum_{c=1}^{k} \sum_{x_i \in C_c} d(x_i, \mu_c)^2$$



(a) Well-separated clusters ($k = 3$) (b) Poorly-separated clusters ($k = 4$)

Fig. 1.6. Clusterings on simple dataset containing three well-separated groups.

When employing Euclidean distance, this is equivalent to the SSE function (1.1) used in the standard k-means algorithm. The *between-cluster sum of squares* is the total of the squares of the distances between the each cluster centroid and the centroid of the entire dataset, denoted $\hat{\mu}$:

$$B(\mathcal{C}) = \sum_{c=1}^{k} |C_c| \; d(\mu_c, \hat{\mu})^2 \quad \text{where} \quad \hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

The statistics $W(\mathcal{C})$ and $B(\mathcal{C})$ have been combined in a number ways by different authors for the purposes of validation. A representative example, the *Calinski-Harabasz* (CH) index [7], involves computing the normalised ratio of within-cluster relative to inter-cluster scatter:

$$CH(\mathcal{C}) = \frac{B(\mathcal{C})/(k-1)}{W(\mathcal{C})/(n-k)} \tag{1.16}$$

A larger value is indicative of greater internal cohesion and a large degree of separation between the clusters in $\mathcal{C}$. This index has been frequently used as a means of automatically selecting the number of cluster in data, particularly in conjunction with agglomerative hierarchical clustering methods.

Generalised Dunn's index

Many popular cluster validation indices are based on the assumption that a correct clustering will minimise intra-cluster dissimilarity, while simultaneously maximising inter-cluster dissimilarity. A prototypical index is that proposed in [18], which was designed to reward "compact and well separated clusters". This index was generalised in [4] to support the use of arbitrary cluster evaluation criteria. Formally, for a disjoint k-way clustering, we let $\Delta : C \rightarrow \mathbb{R}$ denote a function that evaluates the intra-cluster dissimilarity or diameter of a cluster in $\mathcal{C}$, and let $\delta : C \times C \rightarrow \mathbb{R}$ denote a function that

evaluates the inter-cluster dissimilarity between a pair of clusters. An overall evaluation for $\mathcal{C}$ is calculated using the expression:

$$D(\mathcal{C}) = \min_{1 \leq i \leq k} \left\{ \min_{1 \leq j \leq k, i \neq j} \left\{ \frac{\delta(C_i, C_j)}{\max_{1 \leq l \leq k} \{\Delta(C_l)\}} \right\} \right\} \tag{1.17}$$

A larger value for $D(\mathcal{C})$ indicates that the clustering $\mathcal{C}$ consists of compact clusters which are well-separated.

To evaluate clusters, the original formulation of Dunn's index made use of complete intra-cluster diameter and single-linkage inter-cluster distance, as defined by:

$$\Delta_1(C_i) = \max_{x \in C_i, y \in C_i} \{d(x, y)\} \qquad \delta_1(C_i, C_j) = \min_{x \in C_i, y \in C_j} \{d(x, y)\} \tag{1.18}$$

Since both functions only make use of a single distance value corresponding to the most extreme case, this formulation is highly sensitive to the presence of outliers. An alternative approach is to include the contribution of all objects in a cluster by considering object-centroid scatter and measuring inter-cluster dissimilarity in terms of the distance between centroids:

$$\Delta_2(C_i) = 2 \left(\frac{\sum_{x \in C_i} d(x, \mu_i)}{|C_i|} \right) \qquad \delta_2(C_i, C_j) = d(\mu_i, \mu_j) \tag{1.19}$$

Bezdek & Pal [4] suggested that, by considering average object-centroid distance together with average-linkage inter-cluster distance, more robust cluster evaluations can be produced:

$$\Delta_3(C_i) = 2 \left(\frac{\sum_{x \in C_i} d(x, \mu_i)}{|C_i|} \right) \qquad \delta_3(C_i, C_j) = \frac{\sum_{x \in C_i, y \in C_j} d(x, y)}{|C_i|\,|C_j|} \tag{1.20}$$

It should be noted that evaluation criteria based on object-centroid distances will often exhibit an unfair bias toward spherical clusters in the same way as the standard k-means algorithm, leading to the production of misleading results on data where the underlying groups are elongated or non-convex in structure.

Davies-Bouldin index

A related internal validation technique was proposed in [9] that considers the ratio of intra-cluster scatter to inter-cluster separability across all k groups in a clustering. Formally, the *DB index* is defined as a function of the proximity between each cluster and its nearest neighbour:

$$DB(\mathcal{C}) = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} \left\{ \frac{\Delta(C_i) + \Delta(C_j)}{\delta(C_i, C_j)} \right\} \tag{1.21}$$

This value will decrease as clusters become more compact and more distinctly separated, making smaller values for this index desirable. As with Dunn's index, arbitrary cluster evaluation functions can potentially be used in Eqn. 1.21. However, typically the centroid-based metrics defined in Eqn. 1.19 are employed to assess scatter and inter-cluster dissimilarity. A significant disadvantage of the DB index is that it does not have a fixed range, with an output value only constrained to be non-negative, making interpretation problematic. In addition, empirical analysis has shown that, when attempting to select k, this index tends to underestimate the number of groups, particularly for weakly clustered data [15].

C-index

Hubert & Levin [29] proposed a cluster validation measure that evaluates the homogeneity of a set of clusters by comparing the weight of the intra-cluster distances induced to a similar proportion of inter-cluster distances. Formally, let d_w denote the sum of all l_w intra-cluster distances induced by a clustering $\mathcal{C}$. Furthermore, let d_{min} denote the sum of the l_w smallest and d_{max} denote the sum of the l_w largest distances across all pairs of objects in the dataset. Having examined all pairwise distances, the *C-index* for a clustering is calculated as the ratio:

$$HL(\mathcal{C}) = \frac{d_w - d_{min}}{d_{max} - d_{min}} \tag{1.22}$$

A small value for this ratio is generally indicative of a more cohesive clustering.

Silhouette index

Rousseeuw [46] suggested computing a "silhouette value" for each object in a clustering, which measures the degree to which the object belongs to its current cluster relative to the other $k-1$ clusters. Formally, for each $x_i \in C_a$, let $a(i)$ denote the average distance between the object and all other objects in C_a, and let $b(i)$ denote the average distance between x_i and all objects in the nearest competing cluster C_b:

$$a(i) = \frac{1}{|C_a|} \sum_{j \in C_a, i \neq j} d(i,j) \qquad b(i) = \frac{1}{|C_b|} \sum_{j \in C_b} d(i,j)$$

The silhouette width for x_i is then computed using the expression:

$$sil(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{1.23}$$

This produces an evaluation in the range $[-1, 1]$, indicating how well the object fits in its own cluster when compared to how well it would fit if moved

to another cluster. A value close to 1 indicates that x_i is likely to have been assigned to the appropriate cluster, a silhouette closer to 0 suggests that x_i could also have been assigned to the nearest alternative cluster, while a negative value suggests that x_i is likely to have been incorrectly assigned. The latter case can also be interpreted to mean that the object is an outlier. An overall evaluation for a k-way clustering, the *average silhouette width*, can be computed by taking the mean silhouette of all n participating objects:

$$ASW(\mathcal{C}) = \frac{1}{n}\sum_{i=1}^{n} sil(i) \tag{1.24}$$

A higher value for this expression signifies a superior clustering of the data.

1.5.2 External Validation

A significant disadvantage of internal techniques is that useful comparisons may only be made between clusterings that are generated using the same data model and similarity metric [23]. We have also seen that many well-known internal indices make assumptions about the structure of the clusters in data, so that they favour clusters with certain geometric properties.

An alternative approach to validation is to apply the algorithm to a dataset for which a reference partition or "ground truth" is available, typically in the form of predefined class labels. External validation indices make use of this information, unavailable to the clustering algorithm itself, to quantify the level of agreement between the algorithm's output and the set of k' natural classes $\mathcal{C}' = \{C'_1, \ldots, C'_{k'}\}$ in a reference partition. Since these indices generally only consider the final partition of the data, they are independent of the representation and metrics used during the clustering procedure. In this section we provide a comprehensive review of external validation indices that are suitable for evaluating disjoint clusterings. When describing these indices, we let n'_i denote the number of objects in class C'_i, let n_j denote the number of objects in cluster C_j, and let n_{ij} denote the number of objects common to both the class C'_i and cluster C_j

It should be acknowledged that the main role for external measures in machine learning has been in the development and comparative evaluation of clustering algorithms. In the literature it is common for authors to select a fixed value for the number of clusters k, with one or more external indices being subsequently used to gauge the relative merit of the clustering techniques under consideration. In contrast to the common usage of internal indices, it is generally inappropriate to use external criteria to directly select parameters such as k, as this form of *a priori* information is generally inaccessible to the learning algorithm in real applications.

We now provide a review of common external validation indices that may be used to evaluate disjoint clusterings.

Set Matching Measures

A simple external validation approach is to identify a match between each cluster and a corresponding natural class in the reference partition. Once a mapping has been found, evaluations can be readily computed based on the $k' \times k$ *confusion matrix* $\mathbf{N}$, where the entry N_{ij} denotes the size of the intersection $|C'_i \cap C_j|$ between the class C'_i and cluster C_j.

Motivated by conventional evaluation techniques in supervised learning, several authors have suggested assessing the quality of a partition by assigning a unique dominant natural class to each cluster and counting the number of objects that have been assigned to the correct cluster [40]. To do this, a heuristic correspondence procedure is applied, which first identifies the largest intersection N_{ij}, resulting in a match between C'_i and C_j. The next match is chosen based on the highest value N_{ij} from the remaining pairs, with the procedure continuing until $min(k', k)$ matches have been found. Note that no class may be matched to more than one cluster. The *classification accuracy* for the clustering $\mathcal{C}$ is then calculated using the expression

$$H(\mathcal{C}', \mathcal{C}) = \frac{1}{n} \sum_{j'=match(j)} N_{jj'} \tag{1.25}$$

where $match(j)$ denotes the index of the class selected as a match for the cluster C_j.

Zhao & Karypis [56] suggested measuring the extent to which each cluster contains objects from a single dominant natural class. The *purity* of a cluster C_j is defined as the fraction of objects in the cluster that belong to the dominant class contained within that cluster:

$$P(C'_i, C_j) = \frac{1}{n_j} \max_i \{N_{ij}\} \tag{1.26}$$

Unlike the classification accuracy measure, purity allows multiple clusters to be matched to the same dominant class. The overall purity of a clustering is defined as the sum of the individual cluster purities, weighted by the size of each cluster:

$$P(\mathcal{C}', \mathcal{C}) = \sum_{j=1}^{k} \frac{n_j}{n} P(C', C_j) \tag{1.27}$$

This measure provides a naïve estimate of partition quality, where larger purity values are intended to indicate a better clustering. However, the index favours small clusters, with the degenerate case of a singleton cluster resulting in a maximal cluster purity score [51].

The *F-Measure* [37] is based on the *recall* and *precision* criteria that are commonly used in information retrieval tasks. Each cluster is viewed as the result of a query operation, and each natural class is viewed as the target set of documents for the query. In the ideal case, each cluster will directly

correspond to a natural class. Using our notation, precision and recall for a class C'_i and cluster C_j are defined respectively as:

$$p(C_j, C'_i) = \frac{N_{ij}}{n_j} \qquad r(C_j, C'_i) = \frac{N_{ij}}{n'_i}$$

High precision implies that most objects in a given cluster belong to the same class, while high recall suggests that most objects from a single class were assigned to the same cluster. The F-measure for a pair (C'_i, C_j) is given by the harmonic mean of their precision and recall, calculated as:

$$F_{ij} = \frac{2 \cdot r_{ij} \cdot p_{ij}}{p_{ij} + r_{ij}} \tag{1.28}$$

For each class C'_i, a unique matching cluster C_j is selected so as to maximise the value F_{ij}. An overall score for a clustering $\mathcal{C}$ is obtained by taking the weighted average of the maximum F-values across all k' classes:

$$F(\mathcal{C}', \mathcal{C}) = \sum_{i=1}^{k'} \frac{n'_i}{n} \max_j \{F_{ij}\} \tag{1.29}$$

Ghosh [23] notes that this measure tends to favour lower values of k, resulting in coarser clusterings.

Pairwise Co-assignment Measures

An alternative approach to external validation is to count the pairs of objects for which the clusters and natural classes agree on their co-assignment. By considering all pairs, we can calculate statistics for each of four possible cases:

- a = number of pairs in the same class in $\mathcal{C}'$ and assigned to the same cluster in $\mathcal{C}$.
- b = number of pairs in the same class in $\mathcal{C}'$, but in different clusters in $\mathcal{C}$.
- c = number of pairs assigned to the same cluster in $\mathcal{C}$, but in different classes in $\mathcal{C}'$.
- d = number of pairs belonging to different classes in $\mathcal{C}'$ and assigned to different clusters in $\mathcal{C}$.

Note that $a + d$ corresponds to the number of agreements between $\mathcal{C}'$ and $\mathcal{C}$, $b + c$ corresponds to the disagreements, and $M = a + b + c + d = \frac{n(n-1)}{2}$ is the total number of unique pairs.

The *Jaccard coefficient* [30] has been commonly applied to assess the similarity between binary sets. It is also possible for this measure to be used in the context of external validation, where the level of agreement of between the disjoint partitions $\mathcal{C}'$ and $\mathcal{C}$ is given by normalising the number of positive agreements:

$$J(\mathcal{C}', \mathcal{C}) = \frac{a}{a+b+c} \tag{1.30}$$

This index produces a result in the range $[0, 1]$, where a value of 1 indicates that $\mathcal{C}'$ and $\mathcal{C}$ are identical. It was observed in [15] that Eqn. 1.30 tends to produce high values for random clusterings and favours lower values of k.

The *Rand index* [44] is similar to the above measure, but also considers cases where both partitions assign a pair of objects to different groups. This results in an evaluation in the range $[0, 1]$ based on the fraction of pairs for which there is an agreement:

$$R(\mathcal{C}', \mathcal{C}) = \frac{a+d}{a+b+c+d} \tag{1.31}$$

To eliminate biases related to different cluster size distributions and the number of clusters, Hubert & Arabie [28] proposed the *corrected Rand index*, which is computed as follows:

$$CR(\mathcal{C}', \mathcal{C}) = \frac{2(ad-bc)}{(a+b)(b+d)+(a+c)(c+d)} \tag{1.32}$$

After applying this correction, a value of 1 indicates a perfect agreement between the two groupings, while a value of 0 indicates that a clustering is no better than a random partitioning of the data.

Another popular index for assessing the similarity between partitions was proposed in [22], which is based on the calculation of two probability scores: the probability that a pair of objects are assigned to the same cluster given that they belong to the same class, and the probability that a pair objects belong to the same class given that they were assigned to the same cluster. A value for the *Fowlkes-Mallows index* (FM) is found by taking the geometric mean of these probabilities:

$$FM(\mathcal{C}', \mathcal{C}) = \sqrt{\left(\frac{a}{a+b}\right)\left(\frac{a}{a+c}\right)} \tag{1.33}$$

A value close to 1 indicates that the clusters in $\mathcal{C}$ provide a good estimate for the reference partition.

Information Theoretic Measures

Recent research relating to cluster validation has focused on concepts from information theory, which consider the uncertainty of predicting a set of natural classes based on the information provided by a clustering of the same data. We now describe two indices, based on these concepts, which have frequently been applied to evaluate clusterings of text data.

Steinbach *et al.* [50] suggested an entropy-based measure for assessing the agreement between two partitions. By considering the probability $\frac{N_{ij}}{n_j}$ that

an object assigned to cluster C_j belongs to a class C'_i, we can compute the entropy for the assignments in C_j:

$$E(C_j) = -\sum_{i=1}^{k'} \frac{N_{ij}}{n_j} \log \frac{N_{ij}}{n_j} \tag{1.34}$$

An overall score for a clustering $\mathcal{C}$ is given by the sum of the entropy values for each cluster weighted by the fraction of objects assigned to that cluster:

$$E(\mathcal{C}', \mathcal{C}) = \sum_{j=1}^{k} \frac{n_j}{n} E(C_j) \tag{1.35}$$

Smaller values for this measure are desirable, with a value of 0 indicating that each cluster contains instances from a single class. To eliminate the strong bias of Eqn. 1.35 with respect to k, a variant of this index was proposed in [56], where the normalised entropy for a cluster C_j is calculated as:

$$NE(C_j) = -\frac{1}{\log k'} \sum_{i=1}^{k'} \frac{N_{ij}}{n_j} \log \frac{N_{ij}}{n_j} \tag{1.36}$$

Unlike purity and classification accuracy, entropy considers the distribution of all classes in a cluster, rather than a single dominant class. However, this index still exhibits a bias in favour of smaller clusters.

Strehl & Ghosh [51] observed that external measures such as purity and entropy are biased with respect to the number of clusters k, since the probability of each cluster solely containing objects from a single natural class increases as k increases. To address this problem, an alternative index was proposed, based on *mutual information*, which quantifies the amount of information shared between the random variables describing a pair of disjoint partitions.

Formally, let $p'(i)$ and $p(j)$ denote the probabilities that an object belongs to class C'_i and cluster C_j respectively. Furthermore, let $p(i, j)$ denote the joint probability that an object belongs to both C'_i and C_j. For each data object assigned to a class in $\mathcal{C}'$, mutual information evaluates the degree to which knowledge of this assignment reduces the uncertainty regarding the assignment of the object in $\mathcal{C}$. The mean reduction in uncertainty across all objects can be expressed as:

$$I(\mathcal{C}', \mathcal{C}) = \sum_{i=1}^{k'} \sum_{j=1}^{k} p(i, j) \log \frac{p(i, j)}{p'(i)p(j)} \tag{1.37}$$

$I(\mathcal{C}', \mathcal{C})$ takes values between zero and $\min(E(\mathcal{C}'), E(\mathcal{C}))$, where the upper bound is the minimum of the entropy values for the two clusterings. To produce values in the range $[0, 1]$, the authors in [51] proposed *normalised mutual*

information (NMI), where the degree of information shared between the two clusterings is normalised with respect to the geometric mean of their entropies:

$$NI(\mathcal{C}', \mathcal{C}) = \frac{I(\mathcal{C}', \mathcal{C})}{\sqrt{E(\mathcal{C}')E(\mathcal{C})}} \tag{1.38}$$

In practice, an approximation for this quantity, based on cluster assignments, can be calculated using:

$$NMI(\mathcal{C}', \mathcal{C}) = \frac{\sum_{i=1}^{k'} \sum_{j=1}^{k} n_{ij} \log\left(\frac{n \cdot n_{ij}}{n'_i n_j}\right)}{\sqrt{\left(\sum_{i=1}^{k'} n'_i \log\frac{n'_i}{n}\right)\left(\sum_{j=1}^{k} n_j \log\frac{n_j}{n}\right)}} \tag{1.39}$$

An accurate clustering should maximise this score, where a value of 1 indicates an exact correspondence between the assignment of objects in $\mathcal{C}'$ and $\mathcal{C}$, while a value of 0 indicates that knowledge of $\mathcal{C}$ provides no information about the true classes $\mathcal{C}'$. Eqn. 1.39 does have a slight tendency to favour clusterings for larger values of k, although it exhibits no bias against unbalanced cluster sizes.

1.5.3 Stability-based techniques

Recently, a number of methods based on the concept of *stability analysis* have been proposed for the task of model selection. The *stability* of a clustering algorithm refers to its ability to consistently produce similar solutions on data originating from the same source [36]. Since only a single set of data objects will be generally available in unsupervised learning tasks, clusterings are generated on perturbations of the original dataset. A key advantage of stability analysis methods lies in their ability to evaluate a model independently of any specific clustering algorithm or similarity measure. Thus, they represent a robust approach for selecting key algorithm parameters [38].

In this section, we focus on stability-based methods that are relevant when estimating the optimal number of clusters $\hat{k}$ in a dataset. These methods are motivated by the observation that, if the number of clusters in a model is too large, repeated clusterings will lead to arbitrary partitions of the data, resulting in unstable solutions. On the other hand, if the number of clusters is too small, the clustering algorithm will be constrained to merge subsets of objects which should remain separated, also leading to unstable solutions. In contrast, repeated clusterings generated using the optimal number of clusters $\hat{k}$ will generally be consistent, even when the data is perturbed or distorted.

Stability Analysis Based on Resampling

The most common approach to stability analysis involves perturbing the data by randomly sampling the original objects to produce a set of τ non-disjoint

subsets. For each potential value of k in a reasonable range $[k_{min}, k_{max}]$, a corresponding set of τ clusterings are generated on the data subsets. The stability of the clustering model for each candidate value of k is evaluated using indices operating on pairs of hard clusterings, such as the external validation indices described previously. A higher overall stability score suggests that k is a better estimate for the optimal value $\hat{k}$.

A representative example of this approach is the algorithm proposed in [39]. For each value of k, an initial partition $\mathcal{C}_0$ is generated on the entire dataset using a partitional clustering algorithm, which represents a "gold standard" for analysing the stability afforded by using k clusters. Subsequently, τ samples of the data are constructed by randomly selecting a subset of βn data objects without replacement, where $0 \leq \beta \leq 1$ denotes the sampling ratio controlling the number of objects in each sample. A set of clusterings $\{\mathcal{C}_1, \ldots, \mathcal{C}_\tau\}$ is then generated by applying the clustering algorithm to each sample. For each clustering $\mathcal{C}_i$, the fraction of co-assignments preserved from $\mathcal{C}_0$ is calculated, which is equivalent to the Rand index (1.31). An overall evaluation for the stability afforded by k is found by averaging the agreement scores across all τ runs. This process is repeated for each potential $k \in [k_{min}, k_{max}]$. A final estimation for $\hat{k}$ is chosen by identifying the value k leading to the highest average agreement.

Law & Jain [38] proposed an alternative stability analysis approach for model selection where the data is perturbed by bootstrapping. This involves generating τ samples of size n by randomly sampling with replacement. Rather than comparing each clustering to a single gold standard solution, stability is evaluated by considering the level of agreement between each pair of clusterings. A number of indices were considered for assessing agreement, including the Jaccard index (1.30) and the Fowlkes-Mallows index (1.33). The authors note that scores produced by these indices should be corrected for chance to eliminate biases toward smaller values of k. After computing the variance of the corrected agreement scores for each potential value k, the model resulting in the lowest variance is selected as the best estimate for $\hat{k}$.

Ben-Hur *et al.* [2] described a similar approach based on pairwise stability analysis, where agglomerative hierarchical clustering is applied to each sample. By using different cut-off levels from the same hierarchy, the output of a single clustering procedure may be used in the evaluation of all potential values of k. In [24] this approach was extended further to encompass the problem of cluster tendency. This is achieved by setting a threshold θ value for the minimum average pairwise stability that is sufficient to indicate a consistent clustering model. If no stability evaluation exceeds this threshold for any candidate $k \in [k_{min}, k_{max}]$, the data is assumed to have no significant underlying structure. The choice of θ largely depends on the index used to measure the agreement between clusterings.

Prediction-Based Validation

In supervised learning problems, model selection is typically performed by identifying a learning model whose estimated prediction accuracy is highest. A number of authors have suggested that the concept of prediction accuracy can be adapted to the problem of evaluating models in clustering tasks. Recent work by Tibshirani *et al.* [52] has provided a theoretical basis for *prediction-based validation* methods, which assess the stability of a clustering model by measuring the degree to which it allows us to consistently construct a classifier on a training set that will predict the assignment of objects in a clustering of a corresponding test set.

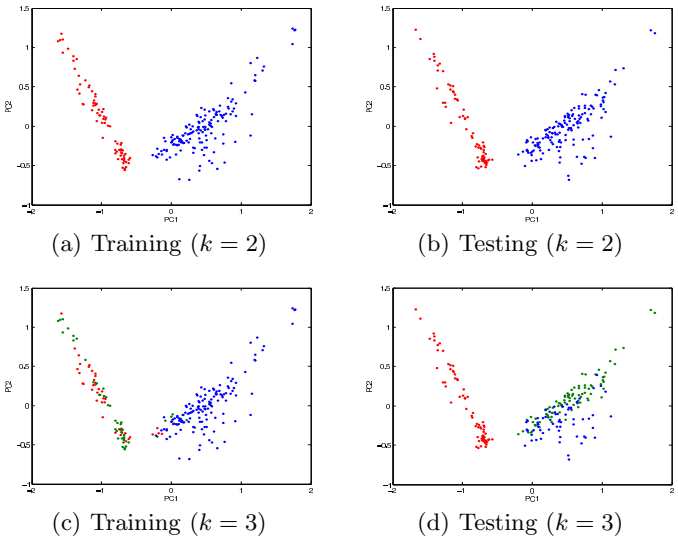
Formally, the validation process involves applying two-fold cross-validation to randomly split a dataset $\mathcal{X}$ into disjoint training and test sets, denoted by $\mathcal{X}_a$ and $\mathcal{X}_b$ respectively. Both sets are then clustered to produce partitions $\mathcal{C}_a$ and $\mathcal{C}_b$, typically using the standard k-means clustering algorithm. Subsequently, a prediction $\mathcal{P}_b$ for the assignment of objects in the test set is produced by assigning each $x_i \in \mathcal{X}_b$ to the nearest centroid in $\mathcal{C}_a$. Prediction accuracy is measured by evaluating the degree to which the class memberships in $\mathcal{P}_b$ correspond to the cluster assignments in $\mathcal{C}_b$.

To numerically evaluate prediction accuracy, a new pairwise measure for comparing partitions was proposed in [52], referred to as *prediction strength*. For each cluster in the test clustering $\mathcal{C}_b = \{C_1, \ldots, C_k\}$, we identify the number of pairs of objects assigned to the same cluster that also belong to the same class in the prediction $\mathcal{P}_b$. These associations can be represented as a $\frac{n}{2} \times \frac{n}{2}$ binary matrix $\mathbf{M}$, where $M_{ij} = 1$ only if the pair (x_i, x_j) are co-assigned in both $\mathcal{C}_b$ and $\mathcal{P}_b$. From this matrix, an evaluation is computed based on the cluster containing the smallest fraction of correctly predicted pairs:

$$S(\mathcal{C}_b, \mathcal{P}_b) = \min_{1 \leq h \leq k} \left[\frac{1}{|C_h|\,(|C_h| - 1)} \sum_{x_i \neq x_j \in C_h} M_{ij} \right] \tag{1.40}$$

The cross-validation process is repeated over τ runs for each candidate value k in the range $[k_{min}, k_{max}]$. The authors suggest a heuristic approach to select the final number of clusters, which is chosen to be the largest k such that $ps(k)$ is above a user-defined threshold. This can be viewed as the selection of the largest number clusters that can be reliably predicted for a given dataset. They note that a threshold in the range $[0.8, 0.9]$ was appropriate for the datasets with which they evaluated the algorithm.

As a simple example, we consider a single cross-validation run for $k = 2$ applied to the set of 34 data objects shown in Figure 7(a). This dataset is randomly divided into two subsets containing 17 objects each. A training clustering $\mathcal{C}_a$ is generated on the first subset and a test clustering $\mathcal{C}_b$ is generated on the second, as shown in figures 7(b) and 7(c) respectively. The centroids μ_1 and μ_2 of $\mathcal{C}_a$ are subsequently used to build a nearest centroid classifier, which produces the predicted classification $\mathcal{P}_b$ for the set of test objects as



(a) Training ($k = 2$) (b) Testing ($k = 2$)

(c) Training ($k = 3$) (d) Testing ($k = 3$)

Fig. 1.7. Example of applying prediction-based validation to examine the suitability of a clustering model with $k = 2$ for a synthetic dataset of 34 data objects.

illustrated in Figure 7(d). By constructing a 17×17 co-assignment matrix $\mathbf{M}$ from $\mathcal{C}_b$ and $\mathcal{P}_b$, and applying Eqn. 1.40, we can calculate that this run leads to a prediction strength of $S(\mathcal{C}_b, \mathcal{P}_b) = 0.43$, indicating that the clustering model is relatively unstable. In practice, multiple cross-validation runs would be applied to produce a result that is robust to the effects of unbiased random sampling.

1.6 Feature Transformation Techniques

1.7 Feature Selection Techniques

References

1. M.A. Aizerman, E.M. Braverman, and L.I. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25(6):821–837, 1964.
2. A. Ben-Hur, A. Elisseeff, and I. Guyon. A stability based method for discovering structure in clustered data. In *Proceedings of the 7th Pacific Symposium on Biocomputing (PSB 2002)*, pages 6–17, Lihue, Hawaii, January 2002.
3. Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien. Using linear algebra for intelligent information retrieval. Technical Report UT-CS-94-270, Computer Science Dept., University of Tennessee, 1994.
4. James C. Bezdek and Nikhil R. Pal. Cluster validation with generalized dunn's indices. In *ANNES '95: Proceedings of the 2nd New Zealand Two-Stream International Conference on Artificial Neural Networks and Expert Systems*, page 190, Washington, DC, USA, 1995. IEEE Computer Society.

5. N. Bolshakova and F. Azuaje. Cluster validation techniques for genome expression data. Technical Report TCD-CS-2002-33, Trinity College Dublin, September 2002.
6. Matthew Brand and Kun Huang. A unifying theorem for spectral embedding and clustering. In *Proc. 9th Int. Workshop on AI and Statistics*, January 2003.
7. T. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3:1–27, 1974.
8. Nello Cristianini and John Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, New York, NY, USA, 2000.
9. D. L. Davies and W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979.
10. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.
11. Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Knowledge Discovery and Data Mining*, pages 269–274, 2001.
12. Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556. ACM Press, 2004.
13. Chris Ding and Xiaofeng He. Cluster merging and splitting in hierarchical clustering algorithms. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, page 139. IEEE Computer Society, 2002.
14. W.E. Donath and A.J. Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Developement*, 17:420–425, 1973.
15. Richard C. Dubes. How many clusters are best? - an experiment. *Pattern Recognition*, 20(6):645–663, 1987.
16. Sandrine Dudoit and Jane Fridlyand. A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome Biology*, 3(7):research0036.1–research0036.21, 2002.
17. J. C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3:32–57, 1974.
18. J. C. Dunn. Well separated clusters and optimal fuzzy-partitions. *Journal of Cybernetics*, 4:95–104, 1974.
19. Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(98):298–305, 1973.
20. Bernd Fischer and Joachim M. Buhmann. Path-based clustering for grouping of smooth curves and texture segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions*, 25(4):513–518, April 2003.
21. E. W. Forgy. Cluster analysis of multivariate data: efficiency vs interpretability of classifications. *Biometrics*, 21:768–769, 1965.
22. E.B. Fowlkes and C.L. Mallow. A method for comparing two hierarchical clusterings. *J. American Statistical Association*, 78:553–569, 1983.
23. J. Ghosh. Scalable clustering methods for data mining. In N. Ye, editor, *Handbook of Data Mining*, chapter 10. Lawrence Erlbaum, 2003.
24. CD Giurcaneanu and I Tabus. Cluster structure inference based on clustering stability with applications to microarray data analysis. *EURASIP Journal on Applied Signal Processing*, 1:64–80, 2004.

25. S. Guha, R. Rastogi, and K. Shim. CURE: an efficient clustering algorithm for large databases. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 73–84, 1998.
26. Kenneth M. Hall. An r-dimensional quadratic placement algorithm. *Managment Science*, 17(3):219–229, November 1970.
27. Ville Hautamäki, Svetlana Cherednichenko, Ismo Kärkkäinen, Tomi Kinnunen, and Pasi Fränti. Improving k-means by outlier removal. In *Image Analysis, 14th Scandinavian Conference, SCIA 2005*, pages 978–987, 2005.
28. L. J. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.
29. L.J. Hubert and J.R. Levin. A general statistical framework for accessing categorical. *Psychological Bulletin*, 83:1072–1082, 1976.
30. P. Jaccard. The distribution of flora in the alpine zone. *New Phytologist*, 11(2):37–50, 1912.
31. Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.
32. L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data*. Wiley & Sons, New York, 1990.
33. B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–308, 1970.
34. Y. Kluger, R. Basri, J.T. Chang, and M. Gerstein. Spectral biclustering of microarray data: Coclustering genes and conditions. *Genome Res.*, 13:703–716, April 2003.
35. H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quaterly*, 2:83–97, 1955.
36. Tilman Lange, Volker Roth, Mikio L. Braun, and Joachim M. Buhmann. Stability-based validation of clustering solutions. *Neural Comput.*, 16(6):1299–1323, 2004.
37. Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22, New York, NY, USA, 1999. ACM Press.
38. Martin Law and Anil K. Jain. Cluster validity by bootstrapping partitions. Technical Report MSU-CSE-03-5, University of Washington, February 2003.
39. Erel Levine and Eytan Domany. Resampling method for unsupervised estimation of cluster validity. *Neural Computation*, 13(11):2573–2593, 2001.
40. Marina Meila. Comparing clusterings. Technical Report 418, University of Washington, 2002.
41. G.W. Milligan and M.C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 1985.
42. A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proc. Advances in Neural Information Processing*, 2001.
43. Alex Pothen, Horst D. Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Mathematical Analysis and Applications*, 11(3):430–452, 1990.
44. W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(66):846–850, 1971.
45. V. Roth, M. Braun, T. Lange, and J. Buhmann. A resampling approach to cluster validation. In *Proceedings of the 15th Symposium in Computational Statistics*, 2002.