Chapter 4

A Survey of Partitional and Hierarchical Clustering Algorithms

Chandan K. Reddy

Department of Computer Science
Wayne State University
Detroit, MI
`reddy@cs.wayne.edu`

Bhanukiran Vinzamuri

Department of Computer Science
Wayne State University
Detroit, MI
`bhanukiranv@wayne.edu`

4.1   Introduction

The two most widely studied clustering algorithms are partitional and hierarchical clustering. These algorithms have been heavily used in a wide range of applications primarily due to their simplicity and ease of implementation relative to other clustering algorithms. Partitional clustering algorithms aim to discover the groupings present in the data by optimizing a specific objective function and iteratively improving the quality of the partitions. These algorithms generally require certain user parameters to choose the prototype points that represent each cluster. For this reason they are also called as *prototype based clustering* algorithms.

Hierarchical clustering algorithms, on the other hand, approach the problem of clustering by developing a binary tree based data structure called the dendrogram. Once the dendrogram is constructed, one can automatically choose the right number of clusters by splitting the tree at different levels to obtain different clustering solutions for the same dataset without re-running the clustering algorithm again. Hierarchical clustering can be achieved in two different ways, namely, bottom-up and top-down clustering. Though, both of these approaches utilize the concept of dendrogram while clustering the data, they might yield entirely different set of results depending on the criterion used during the clustering process.

Partitional methods need to be provided with a set of initial seeds (or clusters) which are then improved iteratively. Hierarchical methods, on the other hand, can start off with the individual data points in single clusters and build the clustering. The role of the distance metric is also different in both of these algorithms. In hierarchical clustering, the distance metric is initially applied on the data points at the base level and then progressively applied on subclusters by choosing *absolute* representative points for the subclusters. However, in the case of partitional methods, in general, the representative points chosen at different iterations can be *virtual* points such as the centroid of the cluster (which is non-existent in the data).

This chapter is organized as follows. In section 4.2, the basic concepts of partitional clustering are introduced and the related algorithms in this field are also discussed. More specifically, subsections 4.2.1-4.2.3 will discuss the widely studied K-Means clustering algorithm and highlight the major factors involved in these partitional algorithms such as initialization methods and estimating the number of clusters K. Subsection 4.2.4 will highlight several variations of the K-Means clustering. The distinctive features of each of these algorithms and their advantages are also highlighted. In section 4.3 the fundamentals of hierarchical clustering are explained. Subsections 4.3.1 and 4.3.2 will discuss the agglomerative and divisive hierarchical clustering algorithms respectively. We will also highlight the differences between the algorithms in both of these categories in this section. Subsection 4.3.3 will briefly discuss the other prominent hierarchical clustering algorithms. Finally, in section 4.4, we will conclude our discussion highlighting the merits and drawbacks of both families of clustering algorithms.

4.2 Partitional Clustering Algorithms

The first partitional clustering algorithm that will be discussed in this section is the *K*-Means clustering algorithm. It is one of the simplest and most efficient clustering algorithms proposed in the literature of data clustering. After the algorithm is described in detail, some of the major factors that influence the final clustering solution will be highlighted. Finally, some of the widely used variations of K-Means will also be discussed in this section.

4.2.1 *K*-Means Clustering

K-means clustering [33, 32] is the most widely used partitional clustering algorithm. It starts by choosing *K* representative points as the initial centroids. Each point is then assigned to the closest centroid based on a particular proximity measure chosen. Once the clusters are formed, the centroids for each cluster are updated. The algorithm then iteratively repeats these two steps until the centroids do not change or any other alternative relaxed convergence criterion is met. *K*-means clustering is a *greedy* algorithm which is guaranteed to converge to a local minimum but the minimization of its score function is known to be NP-Hard [35]. Typically, the convergence condition is relaxed and a weaker condition may be used. In practice, it follows the rule that the iterative procedure must be continued until 1% of the points change their cluster memberships. A detailed proof of the mathematical convergence of *K*-means can be found in [45].

Algorithm 13 *K*-Means Clustering

1: Select *K* points as initial centroids.
2: **repeat**
3:     Form *K* clusters by assigning each point to its closest centroid.
4:     Recompute the centroid of each cluster.
5: **until** Convergence criterion is met.

Algorithm 13 provides an outline of the basic *K*-Means algorithm. Figure 4.1 provides an illustration of the different stages of the running of **3**-means algorithm on the Fisher Iris dataset. The first iteration initializes three random points as centroids. In subsequent iterations the centroids change positions until convergence. A wide range of proximity measures can be used within the *K*-means algorithm while computing the closest centroid. The choice can significantly affect the centroid assignment and the quality of the final solution. The different kinds of measures which can be used here are Manhattan distance (L_1 norm), Euclidean distance (L_2 norm) and Cosine similarity. In general, for the *K*-means clustering, Euclidean distance metric is the most popular choice. As mentioned above, we can obtain different clusterings for different values of *K* and proximity measures. The objective function which is employed by *K*-means is called the Sum of Squared Errors (SSE) or Residual Sum of Squares (RSS). The mathematical formulation for SSE/RSS is provided below.

Given a dataset $D=\{x_1, x_2, \ldots, x_N\}$ consists of *N* points and let us denote the clustering obtained after applying *K*-means clustering by $C = \{C_1, C_2, \ldots, C_k \ldots, C_K\}$. The SSE for this clustering is defined in the Equation (4.1) where c_k is the centroid of cluster C_k. The objective is to find a clustering that minimizes the SSE score. The iterative assignment and update steps of the *K*-means algorithm aim to minimize the SSE score for the given set of centroids.

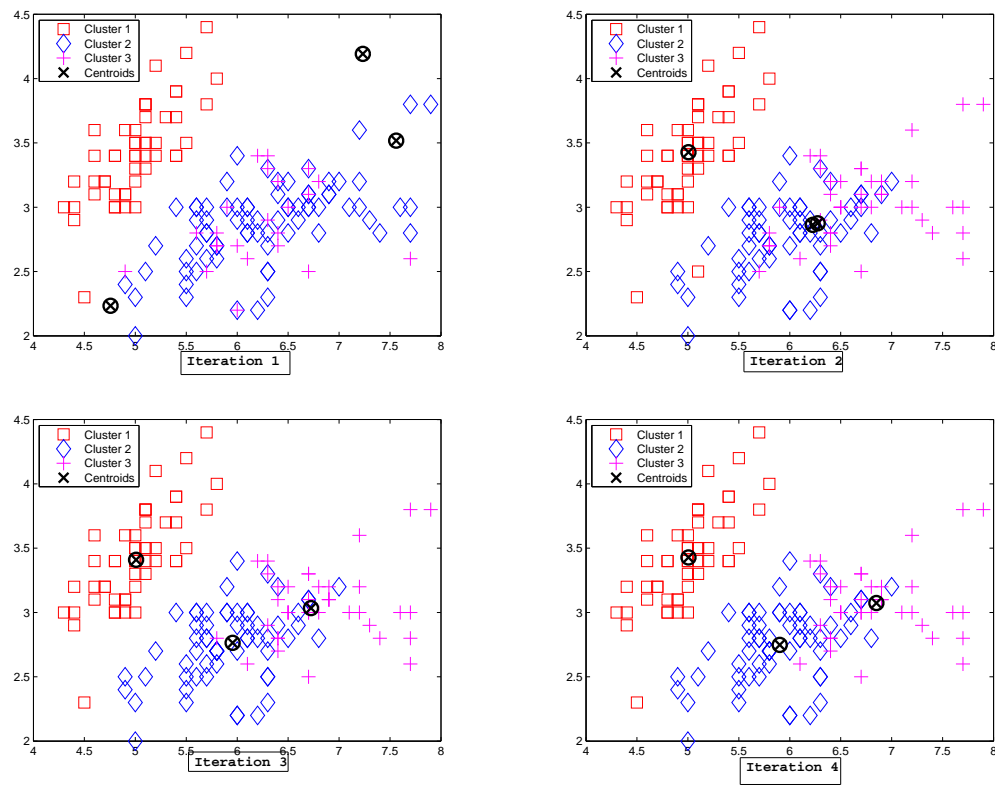$$SSE(C) = \sum_{k=1}^{K} \sum_{x_i \in C_k} \| x_i - c_k \|^2 \tag{4.1}$$

FIGURE 4.1: An illustration of 4 iterations of *K*-means over the Fisher Iris dataset.

$$c_k = \frac{\sum_{x_i \in C_k} x_i}{|C_k|} \tag{4.2}$$

4.2.2 Minimization of Sum of Squared Errors

K-means clustering is essentially an optimization problem with the goal of minimizing the Sum of Squared Error (SSE) objective function. We will mathematically prove the reason behind choosing the mean of the data points in a cluster as the prototype representative for a cluster in the K-means algorithm. Let us denote C_k as the kth cluster, x_i is a point in C_k and c_k is the mean of the kth cluster. We can solve for the representative of C_j which minimizes the SSE by differentiating the SSE with respect to c_j and setting it equal to zero.

$$SSE(C) = \sum_{k=1}^{K} \sum_{x_i \in C_k} (c_k - x_i)^2 \tag{4.3}$$

$$
\begin{aligned}
\frac{\partial}{\partial c_j} SSE &= \frac{\partial}{\partial c_j} \sum_{k=1}^{K} \sum_{x_i \in C_k} (c_k - x_i)^2 \\
&= \sum_{k=1}^{K} \sum_{x_i \in C_j} \frac{\partial}{\partial c_j} (c_j - x_i)^2 \\
&= \sum_{x_i \in C_j} 2 * (c_j - x_i) = 0 \\
\sum_{x_i \in C_j} 2 * (c_j - x_i) &= 0 \Rightarrow |C_j| \cdot c_j = \sum_{x_i \in C_j} x_i \Rightarrow c_j = \frac{\sum_{x_i \in C_j} x_i}{|C_j|}
\end{aligned}
$$

Hence, the best representative for minimizing the SSE of a cluster is the mean of the points in the cluster. In K-means, the SSE monotonically decreases with each iteration. This monotonically decreasing behaviour will eventually converge to a local minimum.

4.2.3 Factors Affecting K-Means

The major factors that can impact the performance of the K-means algorithm are the following:

1. Choosing the initial centroids.

2. Estimating the number of clusters K.

We will now discuss several methods proposed in the literature to tackle each of these factors.

4.2.3.1 Popular Initialization Methods

In his classical paper [33], Macqueen proposed a simple initialization method which chooses K seeds at random. This is the most simplest method and has been widely used in the literature. The other popular K-means initialization methods which have been successfully used to improve the clustering performance are given below.

1. *Hartigan and Wong* [19]: Using the concept of nearest neighbour density, this method suggests that the points which are well-separated and have a large number of points within their surrounding multi-dimensional sphere can be good candidates for initial points. The average

pair-wise Euclidean distance between points is calculated using Equation (4.4). Subsequent points are chosen in the order of their decreasing density and simultaneously maintaining the separation of d_1 from all previous seeds. Note that for the formulae provided below we continue using the same notation as introduced earlier.

$$d_1 = \frac{1}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \| x_i - x_j \| \tag{4.4}$$

2. *Milligan* [37]: Using the results of agglomerative hierarchical clustering (with the help of the dendrogram), this method uses the results obtained from the Ward's method. Ward's method chooses the initial centroids by using the sum of squared errors to evaluate the distance between two clusters. Ward's method is a greedy approach and keeps the agglomerative growth as small as possible.

3. *Bradley and Fayyad* [5]: Choose random subsamples from the data and apply *K*-means clustering to all these subsamples using random seeds. The centroids from each of these subsamples are then collected and a new dataset consisting of only these centroids is created. This new dataset is clustered using these centroids as the initial seeds. The minimum SSE obtained guarantees the best seed set chosen.

4. *K-Means++* [1]: The *K*-means++ algorithm carefully selects the initial centroids for *K*-means clustering. The algorithm follows a simple probability based approach where initially the first centroid is selected at random. The next centroid selected is the one which is farthest from the currently selected centroid. This selection is decided based on a weighted probability score. The selection is continued until we have *K* centroids and then *K*-means clustering is done using these centroids.

4.2.3.2   Estimating the Number of Clusters

The problem of estimating the correct number of clusters (K) is one of the major challenges for the *K*-means clustering. Several researchers have proposed new methods for addressing this challenge in the literature. We will briefly describe some of the most prominent methods.

1. *Calinski-Harabasz Index* [6]: The *Calinski-Harabasz* index is defined by Equation (4.5).

$$CH(K) = \frac{\frac{B(K)}{(K-1)}}{\frac{W(K)}{N-K}} \tag{4.5}$$

where N represents the number of data points. The number of clusters are chosen by maximizing the function given in Equation (4.5). Here $B(K)$ and $W(K)$ are the between and within cluster sum of squares respectively (with K clusters).

2. *Gap Statistic* [48]: In this method, B different datasets each with the same range values as the original data are produced. The within cluster sum of squares is calculated for each of them with different number of clusters. $W_b^*(K)$ is the within cluster sum of squares for the bth uniform dataset.

$$Gap(K) = \frac{1}{B} \times \sum_b log(W_b^*(K)) - log(W(K)) \tag{4.6}$$

where s_k represents the estimate of standard deviation of $log(W_b^*(K))$. The number of clusters chosen is the smallest value of K which satisfies Equation (4.7).

$$Gap(K) \geq Gap(K+1) - s_{k+1} \tag{4.7}$$

3. *Akaike Information Criterion (AIC)* [52]: AIC has been developed by considering the log-likelihood and adding additional constraints of Minimum Description Length (MDL) to estimate the value of K. M represents the dimensionality of the dataset. SSE (Eq.4.1) is the sum of squared errors for the clustering obtained using K. K-means uses a modified AIC as given below.

$$KMeans_{AIC} : K = argmin_K[SSE(K) + 2MK] \tag{4.8}$$

4. *Bayesian Information Criterion (BIC)* [39]: BIC serves as an asymptotic approximation to a transformation of the Bayesian posterior probability of a candidate model. Similar to AIC, the computation is also based on considering the logarithm of likelihood (L). N represents the number of points. The value of K that minimizes the BIC function given below will be used as the initial parameter for running the K-means clustering.

$$BIC = \frac{-2 * ln(L)}{N} + \frac{K * ln(N)}{N} = \frac{1}{N} \times ln(\frac{N^K}{L^2}) \tag{4.9}$$

5. *Duda and Hart* [11]: It is a method for estimation that involves stopping the hierarchical clustering process by choosing the correct cut-off level in the dendrogram. The methods which are typically used to cut a dendrogram are the following: (i) Cutting the dendrogram at a pre-specified level of similarity where the threshold has been specified by the user; (ii) Cutting the dendrogram where the gap between two successive merges is the largest. (iii) Stopping the process when the density of the cluster that results from the merging is below a certain threshold.

6. *Silhouette Coefficient* [26]: This is formulated by considering both the intra- and inter-cluster distances. For a given point x_i, firstly the average of the distances to all points in the same cluster is calculated. This value is set to a_i. Then for each cluster that does not contain x_i, average distance of x_i to all the data points in each cluster is computed. This value is set to b_i. Using these two values the silhouette coefficient of a point is estimated. The average of all the silhouettes in the dataset is called the average silhouettes width for all the points in the dataset. To evaluate the quality of a clustering one can compute the average silhouette coefficient of all points.

$$S = \frac{\sum_{i=1}^{N} \frac{b_i - a_i}{max(a_i, b_i)}}{N} \tag{4.10}$$

7. *Newman and Girvan* [40]: In this method, the dendrogram is viewed as a graph and a betweenness score (which will be used as a dissimilarity measure between the edges) is proposed. The procedure starts by calculating the betweenness score of all the edges in the graph. Then the edge with the highest betweenness score is removed. This is followed by recomputing the betweenness scores between the remaining edges until the final set of connected components are obtained. The cardinality of the set derived through this process serves as a good estimate for K.

8. *ISODATA* [2]: ISODATA was proposed for clustering the data based on the nearest centroid method. In this method, firstly K-means is run on the dataset to obtain the clusters. Clusters are merged if their distance is less than a threshold ϕ or if they have fewer than a certain number of points. Similarly, a cluster is split if the within cluster standard deviation exceeds that of a user defined threshold.

4.2.4 Variations of *K*-Means

The simple framework of the *K*-Means algorithm makes it very flexible to modify and build further more efficient algorithms on top of it. Some of the variations proposed to the *K*-Means algorithm are based on: (i) Choosing different representative prototypes for the clusters (*K*-Medoids, *K*-Medians, *K*-Modes), (ii) Choosing better initial centroid estimates (Intelligent *K*-means, Genetic *K*-means), (iii) Applying some kind of feature transformation techniques (Weighted *K*-Means, Kernel *K*-Means). In this section, we will discuss the most prominent variants of *K*-Means clustering that have been proposed in the literature of partitional clustering.

4.2.4.1 *K*-Medoids Clustering

K-medoids is a clustering algorithm which is more resilient to outliers compared to *K*-means [38]. Similar to *K*-means, the goal of *K*-medoids is also to find a clustering solution that minimizes a pre-defined objective function. The *K*-medoids algorithm chooses the actual data points as the prototypes and is more robust to noise and outliers in the data. The *K*-medoids algorithm aims to minimize the absolute error criterion rather than the SSE. Similar to the *K*-means clustering algorithm, the *K*-medoids algorithm also proceeds iteratively until each representative object is actually the medoid of the cluster. The basic *K*-medoids clustering algorithm is given in Algorithm 14.

In the *K*-medoids clustering algorithm, specific cases are considered where an arbitrary random point x_i is used to replace a representative point m. Following this step the change in the membership of the points that originally belonged to m are checked. The change in membership of these points can occur in one of the two ways. These points can now be closer to x_i (new representative point) or can be closer to any of the other set of representative points. The cost of swapping is calculated as the absolute error criterion for *K*-medoids. For each re-assignment operation this cost of swapping is calculated and this contributes to the overall cost function.

Algorithm 14 *K*-Medoids Clustering

1: Select *K* points as the initial representative objects.
2: **repeat**
3:    Assign each point to the cluster with the nearest representative object.
4:    Randomly select a non-representative object x_i.
5:    Compute the total cost S of swapping the representative object m with x_i.
6:    If $S < 0$, then swap m with x_i to form the new set of *K* representative objects.
7: **until** Convergence criterion is met.

To deal with the problem of executing multiple swap operations while obtaining the final representative points for each cluster, a modification of the *K*-Medoids clustering called Partitioning Around Medoids (PAM) algorithm is proposed [26]. This algorithm operates on the dissimilarity matrix of a given dataset. PAM minimizes the objective function by swapping all the non-medoid points and medoids iteratively until convergence. *K*-Medoids is more robust compared to *K*-means but the computational complexity of *K*-Medoids is higher and hence is not suitable for large datasets. PAM was also combined with a sampling method to propose the Clustering LARge Application (CLARA) algorithm. CLARA considers many samples and applies PAM on each one of them to finally return the set of optimal medoids.

4.2.4.2 *K*-Medians Clustering

The *K*-medians clustering calculates the median for each cluster as opposed to calculating the mean of the cluster (as done in *K*-means). *K*-Medians clustering algorithm chooses *K* cluster centers that aim to minimize the sum of a distance measure between each point and the closest cluster center. The distance measure used in the *K*-medians algorithm is the L_1-norm as opposed to the square of

the L_2-norm used in the K-means algorithm. The criterion function for the K-medians algorithm is defined as follows:

$$S = \sum_{k=1}^{K} \sum_{x_i \in C_k} |x_{ij} - med_{kj}| \tag{4.11}$$

where x_{ij} represents the jth attribute of the instance x_i and med_{kj} represents the median for the jth attribute in the kth cluster C_k. K-medians is more robust to outliers compared to K-means. The goal of the K-Medians clustering is to determine those subset of median points which minimize the cost of assignment of the data points to the nearest medians. The overall outline of the algorithm is similar to that of K-means. The two steps that are iterated until convergence are: (i) All the data points are assigned to their nearest median. (ii) The medians are recomputed using the median of the each individual feature.

4.2.4.3 K-Modes Clustering

One of the major disadvantages of K-means is its inability to deal with non-numerical attributes [51, 3]. Using certain data transformation methods, categorical data can be transformed into new feature spaces and then the K-means algorithm can be applied to this newly transformed space to obtain the final clusters. However, this method has proven to be very ineffective and does not produce good clusters. It is observed that the SSE function and the usage of the mean are not appropriate when dealing with categorical data. Hence, the K-modes clustering algorithm [21] has been proposed to tackle this challenge.

K-modes is a non-parametric clustering algorithm suitable for handling categorical data and optimizes a matching metric (L_0 loss function) without using any explicit distance metric. The loss function here is a special case of the standard L_p norm where p tends to zero. As opposed to the L_p norm which calculates the distance between the data point and centroid vectors, the loss function in K-modes clustering works as a *metric* and uses the number of mismatches to estimate the similarity between the data points. The K-modes algorithm is described in detail in Algorithm 15. As with K-means, this is also an optimization problem and this method also cannot guarantee a global optimal solution.

Algorithm 15 K-Modes Clustering

1: Select K initial modes.
2: **repeat**
3:    Form K clusters by assigning all the data points to the cluster with the nearest mode using the matching metric.
4:    Recompute the modes of the clusters.
5: **until** Convergence criterion is met.

4.2.4.4 Fuzzy K-means Clustering

It is also popularly known as Fuzzy C-Means clustering. Performing hard assignments of points to clusters is not feasible in complex datasets where there are overlapping clusters. To extract such overlapping structures, fuzzy clustering algorithm can be used. In fuzzy C-means clustering algorithm (FCM) [12, 4], the membership of points to different clusters can vary from 0 to 1. The SSE function for FCM is provided in Equation (4.12).

$$SSE(C) = \sum_{k=1}^{K} \sum_{x_i \in C_k} w_{xik}^{\beta} \| x_i - c_k \|^2 \tag{4.12}$$

$$w_{xik} \quad = \quad \frac{1}{\sum\limits_{j=1}^{K} \left(\frac{x_i - c_k}{x_i - c_j}\right)^{\frac{2}{\beta-1}}} \tag{4.13}$$

$$c_k \quad = \quad \frac{\sum\limits_{x_i \in C_k} w_{xik}^{\beta} x_i}{\sum\limits_{x_i \in C_k} w_{xik}} \tag{4.14}$$

Here w_{xik} is the membership weight of point x_i belonging to C_k. This weight is used during the update step of fuzzy *C*-means. The weighted centroid according to the fuzzy weights for C_k is calculated (represented by c_k). The basic algorithm works similar to *K*-means where the algorithm minimizes the SSE iteratively followed by updating w_{xik} and c_k. This process is continued until the convergence of centroids. As in *K*-means, even FCM algorithm is sensitive to outliers and the final solutions obtained will also correspond to the local minimum of the objective function. There are also further extensions of this algorithm in the literature such as *Rough C-means* [34] and *Possibilistic C-means* [30].

4.2.4.5   *X*-Means Clustering

X-Means [42] is a clustering method which can be used to efficiently estimate the value of *K*. It uses a method called *blacklisting* to identify those set of centroids amongst the current existing ones which can be split in order to fit the data better. The decision making here is done using the Akaike or Bayesian Information Criterion. In this algorithm, the centroids are chosen by initially reducing the search space using a heuristic. *K* values for experimentation are chosen between a selected lower and upper bound value. This is followed by assessing the goodness of the model for different *K* in the bounded space using a specific model selection criterion. This model selection criterion is developed using the Gaussian probabilistic model and the maximum likelihood estimates. The best *K* value corresponds to the model that scores the highest on the model score. The primary goal of this algorithm is to estimate *K* efficiently and provide a scalable *K*-Means clustering algorithm when the number of data points becomes large.

4.2.4.6   Intelligent *K*-Means Clustering

Intelligent *K*-means (*IK*-means) clustering [38] is a method which is based on the following principle: *the farther a point is from the centroid the more interesting it becomes*. *IK*-Means uses the basic ideas of principal component analysis (PCA) and selects those points farthest from the centroid which correspond to the maximum data scatter. The clusters derived from such points are called as *anomalous pattern* clusters. The *IK*-means clustering algorithm is given in Algorithm 16.

In Algorithm 16, line 1 initializes the centroid for the dataset as c_g. In line 3, a new centroid is created which is farthest from the centroid of the entire data. In lines 4-5, a version of 2-means clustering assignment is made. This assignment uses the center of gravity of the original dataset cluster c_g and that of the new anomalous pattern cluster s_g as the initial centroids. In line 6, the centroid of the dataset is updated with the centroid of the anomalous cluster. In line 7, a threshold condition is applied to discard small clusters being created because of outlier points. Lines 3-7 are run until one of the stopping criteria is met. (i) Centroids converge or (ii) Pre-specified *K* number of clusters have been obtained or (iii) The entire data has been clustered.

There are different ways by which we can select the *K* for *IK*-Means, some of which are similar to choosing *K* in *K*-means described earlier. A structural based approach which compares the internal cluster cohesion with between-cluster separation can be applied. Standard hierarchical clustering methods which construct a dendrogram can also be used to determine *K*. *K*-means is considered to be a *non-deterministic* algorithm whereas *IK*-means can be considered a *deterministic* algorithm.

IK-Means can be very effective in extracting clusters when they are spread across the dataset

---

Algorithm 16 *IK*-Means Clustering
___
1: Calculate the center of gravity for the given set of data points c_g.
2: **repeat**
3:     Create a centroid c farthest from c_g.
4:     Create a cluster S_{iter} of data points that is closer to c compared to c_g by assigning all the remaining data points x_i to S_{iter} if $d(x_i, c) < d(x_i, c_g)$.
5:     Update the centroid of S_{iter} as s_g.
6:     Set $c_g = s_g$.
7:     Discard small clusters (if any) using a pre-specified threshold.
8: **until** Stopping criterion is met.

---

rather than being compactly structured in a single region. *IK*-means clustering can also be used for initial centroid seed selection before applying *K*-means. At the end of the *IK*-means we will be left with only the good centroids for further selection. Small anomalous pattern clusters will not contribute any candidate centroids as they have already been pruned.

4.2.4.7   Bisecting *K*-Means Clustering

Algorithm 17 Bisecting *K*-Means Clustering
___
1: **repeat**
2:     Choose the parent cluster to be split C.
3:     **repeat**
4:         Select two centroids at random from C.
5:         Assign the remaining points to the nearest subcluster using a pre-specified distance measure.
6:         Recompute centroids and continue cluster assignment until convergence.
7:         Calculate inter-cluster dissimilarity for the 2 subclusters using the centroids.
8:     **until** *I* iterations are completed.
9:     Choose those centroids of the subclusters with maximum inter-cluster dissimilarity.
10:    Split C as C_1 and C_2 for these centroids.
11:    Choose the large cluster among C_1 and C_2 and set it as the parent cluster.
12: **until** *K* clusters have been obtained.

Bisecting *K*-means clustering [47] is a divisive hierarchical clustering method which uses *K*-means repeatedly on the parent cluster *C* to determine the best possible split to obtain two child clusters C_1 and C_2. In the process of determining the best split, bisecting *K*-means obtains uniform sized clusters. The algorithm for Bisecting *K*-means clustering is given in Algorithm 17.

In line 2, the parent cluster to be split is initialized. In lines 4-7, a 2-means clustering algorithm is run *I* times to determine the best split which maximizes the Ward's distance between C_1 and C_2. In lines 9-10, the best split obtained will be used to divide the parent cluster. In line 11, larger among the split clusters is made as the new parent for further splitting. The computational complexity of the Bisecting *K*-means is much higher compared to the standard *K*-Means.

4.2.4.8   Kernel *K*-Means Clustering

In Kernel *K*-means clustering [44], the final clusters are obtained after projecting the data onto the high-dimensional kernel space. The algorithm works by initially mapping the data points in the input space onto a high-dimensional feature space using the kernel function. Some important kernel functions are polynomial kernel, gaussian kernel and sigmoid kernel. The formula for the

SSE criterion of Kernel K-means along with that of the cluster centroid is given in Equation (4.15). The formula for the kernel matrix K for any two points $x_i, x_j \in C_k$ is also given below.

$$SSE(C) = \sum_{k=1}^{K} \sum_{x_i \in C_k} ||\phi(x_i) - c_k||^2 \tag{4.15}$$

$$c_k = \frac{\sum_{x_i \in C_k} \phi(x_i)}{|C_k|} \tag{4.16}$$

$$K_{x_i x_j} = \phi(x_i) \cdot \phi(x_j) \tag{4.17}$$

The difference between the standard K-Means criteria and this new kernel K-means criteria is only in the usage of projection function ϕ. The Euclidean distance calculation between a point and the centroid of the cluster in the high-dimensional feature space in kernel K-means will only require the knowledge of the kernel matrix K. Hence, the clustering can be performed without the actual individual projections $\phi(x_i)$ and $\phi(x_j)$ for the data points x_i, $x_j \in C_k$. It can be observed that the computational complexity is much higher than K-means since the kernel matrix has to be generated from the kernel function for the given data. A weighted version of the same algorithm called *Weighted Kernel K-means* has also been developed [10]. The widely studied spectral clustering can be considered as a variant of kernel K-means clustering.

4.2.4.9 Mean Shift Clustering

Mean shift clustering [7] is a popular non-parametric clustering technique which has been used in many areas of pattern recognition and computer vision. It aims to discover the modes present in the data through a convergence routine. The primary goal of the mean shift procedure is to determine the local maxima or modes present in the data distribution. The Parzen window kernel density estimation method forms the basis for the mean shift clustering algorithm. It starts with each point and then performs a gradient ascent procedure until convergence. As the mean shift vector always points toward the direction of maximum increase in the density, it can define a path leading to a stationary point of the estimated density. The local maxima (or modes) of the density are such stationary points. This mean shift algorithm is one of the widely used clustering methods that fall in the category of mode-finding procedures.

We provide some of the basic mathematical formulation involved in the mean shift clustering algorithm below. Given N data points x_i, where $i = 1, \dots, N$ on a d-dimensional space R^d. Let the multivariate parzen window kernel density estimate $f(x)$ is obtained with kernel $K(x)$ and window radius h. It is given by

$$f(x) = \frac{1}{Nh^d} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right) \tag{4.18}$$

$$m_h(x) = \frac{\sum_{i=1}^{N} x_i \cdot g(|| \frac{x - x_i}{h} ||^2)}{\sum_{i=1}^{N} g(|| \frac{x - x_i}{h} ||^2)} \tag{4.19}$$

More detailed information about obtaining the gradient from the kernel function and the exact kernel functions being used can be obtained from [7]. A proof of convergence of the modes is also provided in [8].

Algorithm 18 Mean Shift Clustering

1: Select K random points as the modes of the distribution.

2: **repeat**

3:   For each given mode x calculate the mean shift vector $m_h(x)$.

4:   Update the point $x = m_h(x)$.

5: **until** Modes become stationary and converge.

4.2.4.10   Weighted K-Means Clustering

Weighted K-Means (WK-Means) algorithm [20] introduces feature weighting mechanism into the standard K-means. It is an iterative optimization algorithm in which the weights for different features are automatically learned. Standard K-means ignores the importance of a particular feature and considers all of the features to be equally important. The modified SSE function optimized by the WK-means clustering algorithm is given in Equation (4.20). Here the features are numbered from $v = 1,\ldots,M$ and the clusters are numbered from $k = 1,\ldots,K$. A user-defined parameter β which employs the impact of the feature weights on the clustering is also used. The clusters are numbered $C = C_1,\ldots,C_k,\ldots,C_K$ and c_k is the M-dimensional centroid for cluster C_k, and c_{kv} represents the vth feature value of the centroid. Feature weights are updated in WK-means according to w_v. D_v is the sum of within cluster variances of feature v weighted by cluster cardinalities.

$$SSE(C,w) = \sum_{k=1}^{K} \sum_{x_i \in C_k} \sum_{v=1}^{M} s_{xik} w_v^{\beta} (x_{iv} - c_{kv})^2 \tag{4.20}$$

$$w_v = \frac{1}{\sum_{u \in V} [\frac{D_v}{D_u}]^{\frac{1}{\beta - 1}}} \tag{4.21}$$

$$d(x_i, c_k) = \sum_{v=1}^{M} w_v^{\beta} (x_{iv} - c_{kv})^2 \tag{4.22}$$

$$\begin{cases} s_{xik} \in (0,1) \\ \sum\limits_{k=1}^{K} s_{xik} = 1 \\ \sum\limits_{v=1}^{M} w_v = 1 \end{cases} \tag{4.23}$$

Algorithm 19 Weighted K-means Clustering

1: Choose K random centroids and set up M feature weights such that they sum to 1.

2: **repeat**

3:   Assign all data points x_i to the closest centroid by calculating $d(x_i, c_k)$.

4:   Recompute centroids of the clusters after completing assignment.

5:   Update weights using w_v.

6: **until** Convergence criterion has been met.

The WK-means clustering algorithm runs similar to K-means clustering but the distance measure is also weighted by the feature weights. In line 1, the centroids and weights for M features are initialized. In lines 3-5, points are assigned to their closest centroids and the weighted centroid is calculated. This is followed by a weight update step such that the sum of weights is constrained as

shown in Equation (4.23). These steps are continued until the centroids converge. This algorithm is computationally more expensive compared to K-Means. Similar to K-means, this algorithm also suffers from convergence issues. Intelligent K-means (IK-Means) [38] can also be integrated with WK-means to yield the Intelligent Weighted K-means algorithm.

4.2.4.11 Genetic K-Means Clustering

K-means suffers from the problem of converging to a local minimum. To tackle this problem, stochastic optimization procedures which are good at avoiding the convergence to a local optimal solution can be applied. Genetic algorithms (GA) are proven to converge to a global optimum. These algorithms evolve over *generations*, where during each generation they produce a new population from the current one by applying a set of genetic operators such as natural selection, crossover, and mutation. They develop a fitness function and pick up the fittest individual based on the probability score from each generation and use them to generate the next population using the mutation operator. The problem of local optima can be effectively solved by using GA and this gives rise to the Genetic K-means algorithm (GKA) [29]. The data is initially converted using a string of group numbers coding scheme and a population of such strings forms the initial dataset for GKA. The following are the major steps involved in the GKA algorithm.

1. **Initialization**: Selecting a random population initially to begin the algorithm. This is analogous to the random centroid initialization step in K-Means.

2. **Selection**: Using the probability computation given in Equation (4.24), identify the fittest individuals in the given population.

$$P(s_i) = \frac{F(s_i)}{\sum_{j=1}^{N} F(s_j)} \qquad (4.24)$$

   where $F(s_i)$ represents the fitness value of a string s_i in the population. A fitness function is further developed to assess the goodness of the solution. This fitness function is analogous to the SSE of K-Means.

3. **Mutation**: This is analogous to the K-Means assignment step where points are assigned to their closest centroids followed by updating the centroids at the end of iteration. The selection and mutation steps are applied iteratively until convergence is obtained.

The pseudocode of the exact GKA algorithm is discussed in detail in [29] and a proof of convergence of the GA is given [43].

4.2.5 Making K-Means Faster

It is believed that the K-Means Clustering algorithm consumes a lot of time in its later stages when the centroids are close to their final locations but the algorithm it yet to converge. An improvement to the original Lloyd's K-Means clustering using a kd-tree data structure to store the data points was proposed in [24]. This algorithm is called the *filtering algorithm* where for each node a set of candidate centroids is maintained similar to a normal kd-tree. These candidate set centroids are pruned based on a distance comparison which measures the proximity to the midpoint of the cell. This filtering algorithm runs faster when the separation between the clusters increases. In the K-Means clustering algorithm, usually there are several redundant calculations that are performed. For example, when a point is very far from a particular centroid, then calculating its distance to that centroid may not be necessary. The same applies for a point which is very close to the centroid as it can be directly assigned to the centroid without computing its exact distance. An optimized

K-Means clustering method which uses the triangle inequality metric is also proposed to reduce the number of distance metric calculations [13]. The mathematical formulation for the lemma used by this algorithm is as follows. Let *x* be a data point and let *b* and *c* be the centroids.

$$d(b,c) \geq 2d(x,b) \rightarrow d(x,c) \geq d(x,b) \tag{4.25}$$

$$d(x,c) \geq max\{0, d(x,b) - d(b,c)\} \tag{4.26}$$

This algorithm runs faster than the standard *K*-Means clustering algorithm as it avoids both kinds of computations mentioned above by using the lower and upper bounds on distances without affecting the final clustering result.

4.3 Hierarchical Clustering Algorithms

Hierarchical clustering algorithms [23] were developed to overcome some of the disadvantages associated with flat or partitional based clustering methods. Partitional methods generally require a user pre-defined parameter *K* to obtain a clustering solution and they are often *non-deterministic* in nature. Hierarchical algorithms were developed to build a more deterministic and flexible mechanism for clustering the data objects. Hierarchical methods can be categorized into *agglomerative* and *divisive* clustering methods. Agglomerative methods start by taking singleton clusters (that contain only one data object per cluster) at the bottom level and continue merging two clusters at a time to build a *bottom-up* hierarchy of the clusters. Divisive methods, on the other hand, start with all the data objects in a huge macro-cluster and split it continuously into two groups generating a *top-down* hierarchy of clusters.

A cluster hierarchy here can be interpreted using the standard binary tree terminology as follows. The root represents all the set of data objects to be clustered and this forms the apex of the hierarchy (level 0). At each level, the child entries (or nodes) which are subsets of the entire dataset correspond to the clusters. The entries in each of these clusters can be determined by traversing the tree from the current cluster node to the base singleton data points. Every level in the hierarchy corresponds to some set of clusters. The base of the hierarchy consists of all the singleton points which are the leaves of the tree. This cluster hierarchy is also called a *dendrogram*. The basic advantage of having a hierarchical clustering method is that it allows for cutting the hierarchy at any given level and obtaining the clusters correspondingly. This feature makes it significantly different from partitional clustering methods in that it does not require a pre-defined user specified parameter *k* (number of clusters). We will discuss more details of how the dendrogram is cut later in this chapter.

In this section, we will first discuss different kinds of agglomerative clustering methods which primarily differ from each other in the similarity measures that they employ. The widely studied algorithms in this category are the following: *single link* (nearest neighbour), *complete link* (diameter), *group average* (average link),*centroid similarity* and *Ward's method* (minimum variance). Subsequently, we will also discuss some of the popular divisive clustering methods.

4.3.1 Agglomerative Clustering

The basic steps involved in an agglomerative hierarchical clustering algorithm are the following. Firstly, using a particular proximity measure a dissimilarity matrix is constructed and all the data points are visually represented at the bottom of the dendrogram. The closest set of clusters are merged at each level and then the dissimilarity matrix is updated correspondingly. This process of agglomerative merging is carried on until the final maximal cluster (that contains all the data objects in a single cluster) is obtained. This would represent the apex of our dendrogram and mark

the completion of the merging process. We will now discuss about the different kinds of proximity measures which can be used in agglomerative hierarchical clustering. Subsequently, we will also provide a complete version of the agglomerative hierarchical clustering algorithm in Algorithm 20.

4.3.1.1 Single and Complete Link

The most popular agglomerative clustering methods are single link and complete link clusterings. In *single link clustering* [36, 46], the similarity of two clusters is the similarity between their most similar (nearest neighbour) members. This method intuitively gives more importance to the regions where clusters are closest and neglecting the overall structure of the cluster. Hence, this method falls under the category of a *local* similarity based clustering method. Because of its local behaviour, single linkage is capable of effectively clustering non-elliptical, elongated shaped groups of data objects. However, one of the main drawbacks of this method is its sensitivity to noise and outliers in the data.

Complete link clustering [27] measures the similarity of two clusters as the similarity of their most dissimilar members. This is equivalent to choosing the cluster pair whose merge has the smallest diameter. As this method takes the cluster structure into consideration it is non-local in behaviour and generally obtains compact shaped clusters. However, similar to single link clustering, this method is also sensitive to outliers. Both single link and complete link clustering have their graph-theoretic interpretations [16], where the clusters obtained after single link clustering would correspond to the connected components of a graph and those obtained through complete link would correspond to the maximal cliques of the graph.
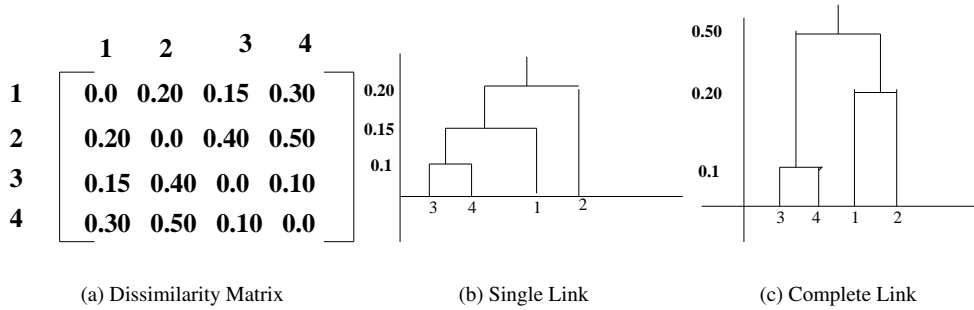

(a) Dissimilarity Matrix   (b) Single Link   (c) Complete Link

FIGURE 4.2: An illustration of agglomerative clustering. (a) A dissimilarity matrix computed for four arbitrary data points. The corresponding dendrograms obtained using (b) single link and (c) complete link hierarchical clustering methods.

$$
\begin{aligned}
d_{min}((3,4),1) &= min(d(3,1),d(4,1)) = 0.15 &\quad (4.27)\\
d_{min}((3,4,1),2) &= min(d(3,2),d(4,2),d(1,2)) = 0.20\\
d_{max}((3,4),1) &= max(d(3,1),d(4,1)) = 0.30\\
d_{max}((3,4),2) &= max(d(3,2),d(4,2)) = 0.50\\
d_{max}((3,4),(1,2)) &= max(d(3,1),d(3,2),d(4,1),d(4,2)) = 0.50
\end{aligned}
$$

Figure 4.2 shows the dissimilarity matrix and the corresponding two dendrograms obtained using single link and complete link algorithms on a toy dataset. In the dendrograms, the X-axis indicates the data objects and the Y-axis indicates the dissimilarity (distance) at which the points were merged. The difference in merges between both the dendrograms occurs due to the different

criteria used by single and complete link algorithms. In single link, firstly data points 3 and 4 are merged at 0.1 as shown in (b). Then, based on the computations shown in Equation (4.27), cluster (3,4) is merged with data point 1 at the next level; at the final level cluster (3,4,1) is merged with 2. In complete link, merges for cluster (3,4) are checked with points 1 and 2 and as d(1,2)=0.20, points 1 and 2 are merged at the next level. Finally, clusters (3,4) and (1,2) are merged at the final level. This explains the difference in the clustering in both the cases.

4.3.1.2 Group Averaged and Centroid Agglomerative Clustering

Group Averaged Agglomerative Clustering (GAAC) considers the similarity between all pairs of points present in both the clusters and diminishes the drawbacks associated with single and complete link methods. Before we look at the formula let us introduce some terminology. Let two clusters C_a and C_b be merged so that the resulting cluster is $C_{a \cup b} = C_a \cup C_b$. The new centroid for this cluster is $c_{a \cup b} = \frac{N_a c_a + N_b c_b}{N_a + N_b}$, where N_a and N_b are the cardinalities of the clusters C_a and C_b respectively. The similarity measure for GAAC is calculated as follows:

$$S_{GAAC}(C_a, C_b) = \frac{1}{(N_a + N_b)(N_a + N_b - 1)} \sum_{i \in C_a \cup C_b} \sum_{j \in C_a \cup C_b, i \neq j} d(i, j) \qquad (4.28)$$

We can see that the distance between two clusters is the average of all the pair-wise distances between the data points in these two clusters. Hence, this measure is expensive to compute especially when the number of data objects becomes large. *Centroid based agglomerative clustering*, on the other hand, calculates the similarity between two clusters by measuring the similarity between their centroids. The primary difference between GAAC and Centroid agglomerative clustering is that, GAAC considers all pairs of data objects for computing the average pair-wise similarity, whereas centroid based agglomerative clustering uses only the centroid of the cluster to compute the similarity between two different clusters.

4.3.1.3 Ward's Criterion

Ward's criterion [49, 50] was proposed to compute the distance between two clusters during agglomerative clustering. This process of using Ward's criterion for cluster merging in agglomerative clustering is also called as *Ward's agglomeration*. It uses the *K*-means squared error criterion to determine the distance. For any two clusters, C_a and C_b, the Ward's criterion is calculated by measuring the increase in the value of the sum of squared error criterion (SSE) for the clustering obtained by merging them into $C_a \cup C_b$. The Ward's criterion is defined as follows:

$$
\begin{aligned}
W(C_{a \cup b}, c_{a \cup b}) - W(C, c) &= \frac{N_a N_b}{N_a + N_b} \sum_{v=1}^{M} (c_{av} - c_{bv})^2 \qquad (4.29) \\
&= \frac{N_a N_b}{N_a + N_b} d(c_a, c_b)
\end{aligned}
$$

So the Ward's criterion can be interpreted as the squared Euclidean distance between the centroids of the merged clusters C_a and C_b weighted by a factor that is proportional to the product of cardinalities of the merged clusters.

4.3.1.4 Agglomerative Hierarchical Clustering Algorithm

In Algorithm 20, we provide a basic outline of an agglomerative hierarchical clustering algorithm. In line 1, the dissimilarity matrix is computed for all the points in the dataset. In lines 3-4, the closest pair of clusters are repeatedly merged in a bottom-up fashion and the dissimilarity matrix is updated. The rows and columns pertaining to the older clusters are removed from the dissimilarity matrix and are added for the new cluster. Subsequently, merging operations are carried out with this updated dissimilarity matrix. Line 5 indicates the termination condition for the algorithm.

Algorithm 20 Agglomerative Hierarchical Clustering

1: Compute the dissimilarity matrix between all the data points.
2: **repeat**
3:     Merge clusters as $C_{a \cup b} = C_a \cup C_b$. Set new cluster's cardinality as $N_{a \cup b} = N_a + N_b$.
4:     Insert a new row and column containing the distances between the new cluster $C_{a \cup b}$ and the remaining clusters.
5: **until** Only one maximal cluster remains

4.3.1.5 Lance-Williams Dissimilarity Update Formula

We discussed many different proximity measures that are used in agglomerative hierarchical clustering. A convenient formulation in terms of dissimilarity which embraces all the hierarchical methods mentioned so far is the Lance-Williams dissimilarity update formula [31]. If points i and j are agglomerated into cluster $i \cup j$, then we will have to just specify the new dissimilarity between the cluster and all other points. The formula is given as follows:

$$d(i \cup j, k) = \alpha_i d(i,k) + \alpha_j d(j,k) + \beta d(i,j) + \gamma |d(i,k) - d(j,k)| \qquad (4.30)$$

Here, α_i, α_j, β, and γ define the agglomerative criterion. These coefficient values for the different kinds of methods we have studied so far are provided in Table 4.1.

TABLE 4.1: Values of the coefficients for the Lance-Williams dissimilarity update formula for different hierarchical clustering algorithms.

Name of the method	Lance Williams Dissimilarity Update formula
Single Link	$\alpha_i = 0.5$; $\beta = 0$; and $\gamma = -0.5$
Complete Link	$\alpha_i = 0.5$; $\beta = 0$; and $\gamma = 0.5$
GAAC	$\alpha_i = \frac{|i|}{|i|+|j|}$; $\beta = 0$; and $\gamma = 0$
Centroid	$\alpha_i = \frac{|i|}{|i|+|j|}$; $\beta = -\frac{|i||j|}{(|i|+|j|)^2}$; and $\gamma = 0$
Ward's	$\alpha_i = \frac{|i|+|k|}{|i|+|j|+|k|}$; $\beta = -\frac{|k|}{|i|+|j|+|k|}$; and $\gamma = 0$

4.3.2 Divisive Clustering

Divisive hierarchical clustering is a *top-down* approach where the procedure starts at the root with all the data points and recursively splits it to build the dendrogram. This method has the advantage of being more efficient compared to agglomerative clustering especially when there is no need to generate a complete hierarchy all the way down to the individual leaves. It can be considered as a *global approach* since it contains the complete information before splitting the data.

4.3.2.1 Issues in Divisive Clustering

We will now discuss the factors that affect the performance of divisive hierarchical clustering.

1. *Splitting criterion*: The Ward's K-means square error criterion is used here. The greater reduction obtained in the difference in the SSE criterion should reflect the goodness of the split. Since the SSE criterion can be applied to numerical data only, Gini index (which is widely used in decision tree construction in classification) can be used for handling the nominal data.

2. *Splitting method*: The splitting method used to obtain the binary split of the parent node is also critical since it can reduce the time taken for evaluating the Ward's criterion. The Bisecting K-means approach can be used here (with K=2) to obtain good splits since it is based on the same criterion of maximizing the Ward's distance between the splits.

3. *Choosing the cluster to split*: The choice of cluster chosen to split may not be as important as the first two factors, but it can still be useful to choose the most appropriate cluster to further split when the goal is to build a compact dendrogram. A simple method of choosing the cluster to be split further could be done by merely checking the square errors of the clusters and splitting the one with the largest value.

4. *Handling noise*: Since the noise points present in the dataset might result in aberrant clusters, a threshold can be used to determine the termination criteria rather splitting the clusters further.

4.3.2.2 Divisive Hierarchical Clustering Algorithm

In Algorithm 21, we provide the steps involved in divisive hierarchical clustering. In line 1, we start with all the points contained in the maximal cluster. In line 3, the Bisecting K-means approach is used to determine the uniform splitting mechanism to obtain C_1 and C_2. In line 4, we use the heuristic mentioned above and choose the cluster with higher squared error for splitting as the next parent. These steps (lines 3-4) are run repeatedly until the complete dendrogram (up to the individual leaves) has been constructed. As mentioned above, we can use the threshold to handle noise during the construction of the dendrogram.

Algorithm 21 Basic Divisive Hierarchical Clustering

1: Start with the root node consisting all the data points
2: **repeat**
3:    Split parent node into two parts C_1 and C_2 using Bisecting K-means to maximize Ward's distance $W(C_1, C_2)$.
4:    Construct the dendrogram. Among the current leaves, choose the cluster with the highest squared error.
5: **until** Singleton leaves are obtained.

4.3.2.3 Minimum Spanning Tree based Clustering

In a weighted graph, a minimum spanning tree is an acyclic subgraph that covers all the vertices with the minimum edge weights. Prim's and Kruskal's algorithms [9] are used for finding the minimum spanning tree (MST) in a weighted graph. In a Euclidean minimum spanning tree (EMST), the data points represent the vertices and the edge weights are computed using the Euclidean distance between two data points. Each edge in an EMST represents the shortest distance between those two points. Using this EMST a divisive clustering method can be developed which removes the largest weighted edge to get two clusterings and subsequently removes the next largest edge to get three clusterings and so on. This process of removing edges from an EMST gives rise to an effective divisive clustering method. The major advantage of this method is that it is able to detect clusters with non-spherical shapes effectively.

A basic EMST clustering algorithm proceeds by taking a user supplied parameter K where the edges present in the graph are sorted in a descending order. This is followed by removing the edges with the top (K-1) weights one by one to get the K connected components. This is similar to the process of divisive clustering where finer clustering partitions are obtained after each split. Subsequently, we can also use the EMST to build a clustering algorithm which continuously prunes

the inconsistent edges present in the graph. An inconsistent edge is the one whose edge weight is much higher than the average weight of the edges in the neighbourhood of that edge. Algorithm 22 describes the minimum spanning tree based clustering algorithm that was originally proposed by Zahn [53].

Algorithm 22 Zahn Minimum Spanning Tree Based Divisive Clustering

---

1: Create the EMST using Prim's/Kruskal's algorithm on all the data points.
2: **repeat**
3:   Remove edge with highest inconsistency measure.
4: **until** No more inconsistent edges can be removed.

4.3.3 Other Hierarchical Clustering Algorithms

The agglomerative and divisive hierarchical clustering methods are successful in capturing convex shaped clusters effectively. As mentioned above, agglomerative methods, especially single link and complete link, suffer from the "chaining problem" and are ineffective at capturing arbitrarily-shaped clusters. Hence, to capture arbitrary shaped clusters, algorithms such as CURE [17] and CHAMELEON [25] have been proposed in the literature. Some of the popular extensions of hierarchical algorithms are discussed below.

1. **CURE** (Clustering Using REpresentatives) [17] is an algorithm which incorporates a novel feature of representing a cluster using a set of well scattered representative points. The distance between two clusters is calculated by looking at the minimum distance between the representative points chosen. In this manner, CURE incorporates features of both the Single link and GAAC hierarchical clustering methods. Choosing scattered points helps CURE capture clusters of arbitrary shapes also. In addition, CURE employs a shrinking factor α in the algorithm, where the points are shrunk towards the centroid by a factor α. α shrinking has a greater effect in the case of outliers compared to normal points. This makes CURE more robust to outliers. Similar to this approach, an algorithm called ROCK [18] was also proposed to handle categorical data. This algorithm uses the concept of common links and determines the Jaccard coefficient between candidate clusters for hierarchical clustering.

2. **CHAMELEON** [25] is a clustering algorithm which uses graph partitioning methods on the K-nearest neighbour graph of the data. These initial partitions are then used as the seed clusters for the agglomerative hierarchical clustering process. The algorithm uses two metrics based on the relative inter-connectivity and relative closeness of clusters to merge the clusters. These metrics capture the local information of the clusters during the clustering process thus enabling this algorithm to behave like a *dynamic* framework. CHAMELEON is one of the best hierarchical clustering algorithms and is extremely effective in capturing arbitrary shaped clusters which is primarily due to the dynamic behaviour. A detailed comparison between the clustering results of CURE and CHAMELEON for synthetic datasets with clusters of varying shapes can also be found in [25].

3. **COBWEB** [15]: This is a conceptual clustering algorithm that works incrementally by updating the clusters object by object. Probabilistically described clusters arranged as a tree to form a hierarchical clustering known as probabilistic categorization tree. It handles uncertainty associated with categorical attributes in clustering through a probabilistic framework that is similar to Naive Bayes. The dendrogram in this algorithm is also called a classification tree and the nodes are referred to as concepts.

4. **Self-Organizing Maps (SOM)** [28] were developed on the same lines of Artificial Neural

Networks and are useful for hierarchical representation. It is an efficient data visualization technique. Similar to *K*-means, data points are assigned to their closest centroids. The difference arises in the centroid update step where when a centroid is updated then those in its neighbourhood which are close to this centroid are also updated. The final output is a SOM neural network which can be explored to understand the relationships between different objects involved in the clustering.

4.4   Discussion and Summary

A major advantage of partitional clustering algorithms is that they can gradually improve the clustering quality through iterative optimization process [3]. This cannot be done in standard hierarchical clustering since the dendrogram cannot revisit the merges (or splits) that were already done. Partitional algorithms are also effective in detecting compact spherical shaped clusters and are easy to implement and use in practice [22]. *K*-means is also a computationally efficient algorithm compared to hierarchical clustering. Although there is no consensus, it is believed that *K*-means is better than hierarchical clustering algorithms [35] in terms of the quality of the final clustering solution.

Hierarchical clustering methods can potentially overcome some of the critical problems associated with flat (partitional) clustering methods. One of the major advantages of hierarchical algorithms is the generation of the visual dendrograms which can assist the end-users during the clustering process. In such applications, generally a user will label the clusters to understand more about them. This is also called as cluster labelling. Hierarchical methods are also *deterministic* compared to the non-deterministic behaviour experienced with the basic *K*-means algorithm.

Despite these advantages, it is observed that in hierarchical clustering methods the merge or split decisions once made at any given level in the hierarchy cannot be undone [3]. This is considered to be a weakness for such hierarchical algorithms since it reduces the flexibility. To overcome this problem, [14] proposes an iterative optimization strategy that keeps modifying the created dendrogram until the optimal solution is obtained. The run-time complexity of these hierarchical algorithms is quadratic which is not desirable especially for large-scale problems. Parallel hierarchical clustering methods [41] have also been proposed to reduce the complexity to linear time.

In spite of the numerous advances made in the field of data clustering in the past two decades, both partitional and hierarchical clustering algorithms form a solid foundation for data clustering. Many of the newly proposed data clustering algorithms (to be discussed in the next few chapters) typically compare their performance to these fundamental clustering algorithms. In addition, due to their simplicity and ease of usage, these algorithms are heavily used in several other application domains such as bioinformatics, information retrieval, text mining, imaging, climate science, and astronomy. The development of new variants of both partitional and hierarchical clustering algorithms is still an active area of research.

Bibliography

[1] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[2] G. H. Ball and D. J. Hall. Isodata, a novel method of data analysis and pattern classification. Technical report, DTIC Document, 1965.

[3] P. Berkhin. A survey of clustering data mining techniques. *Grouping multidimensional data*, pages 25–71, 2006.

[4] J. C. Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Kluwer Academic Publishers, 1981.

[5] P. S. Bradley and U. M. Fayyad. Refining initial points for k-means clustering. In *Proceedings of the Fifteenth International Conference on Machine Learning*, volume 66. San Francisco, CA, USA, 1998.

[6] T. Caliński and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-Theory and Methods*, 3(1):1–27, 1974.

[7] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

[8] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[9] T. H. Cormen. *Introduction to algorithms*. The MIT press, 2001.

[10] I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 551–556. ACM, 2004.

[11] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[12] J. C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[13] C. Elkan. Using the triangle inequality to accelerate k-Means. *In the Proceedings of International Conference on Machine Learning (ICML)*, pages 147–153, 2003.

[14] D. Fisher. Optimization and simplification of hierarchical clusterings. In *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 118–123, 1995.

[15] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.

[16] J. C. Gower and G. J. S. Ross. Minimum spanning trees and single linkage cluster analysis. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 18(1):54–64, 1969.

[17] S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. In *ACM SIGMOD Record*, volume 27, pages 73–84. ACM, 1998.

[18] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering*, pages 512–521. IEEE, 1999.

[19] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[20] J. Z. Huang, M. K. Ng, H. Rong, and Z. Li. Automated variable weighting in k-means type clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):657–668, 2005.

[21] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998.

[22] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[23] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[24] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002.

[25] G. Karypis, E. H. Han, and V. Kumar. CHAMELEON: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[26] L. Kaufman, P.J. Rousseeuw, et al. *Finding groups in data: an introduction to cluster analysis*, volume 39. Wiley Online Library, 1990.

[27] B. King. Step-wise clustering procedures. *Journal of the American Statistical Association*, 62(317):86–101, 1967.

[28] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[29] K. Krishna and M. N. Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 29(3):433–439, 1999.

[30] R. Krishnapuram and J. M. Keller. The possibilistic C-means algorithm: insights and recommendations. *IEEE Transactions on Fuzzy Systems*, 4(3):385–393, 1996.

[31] G. N. Lance and W. T. Williams. A general theory of classificatory sorting strategies ii. clustering systems. *The computer journal*, 10(3):271–277, 1967.

[32] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[33] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297, California, USA, 1967.

[34] P. Maji and S. K. Pal. Rough set based generalized fuzzy c-means algorithm and quantitative indices. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(6):1529–1540, 2007.

[35] C. D. Manning, P. Raghavan, and H. Schutze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[36] L. L. McQuitty. Elementary linkage analysis for isolating orthogonal and oblique types and typal relevancies. *Educational and Psychological Measurement; Educational and Psychological Measurement*, 1957.

[37] G. W. Milligan. A monte carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika*, 46(2):187–199, 1981.

[38] B. G. Mirkin. *Clustering for data mining: a data recovery approach*, volume 3. CRC Press, 2005.

[39] R. Mojena. Hierarchical grouping methods and stopping rules: an evaluation. *The Computer Journal*, 20(4):359–363, 1977.

[40] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113+, 2003.

[41] C. F. Olson. Parallel algorithms for hierarchical clustering. *Parallel computing*, 21(8):1313–1325, 1995.

[42] D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734, San Francisco, 2000.

[43] G. Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1):96–101, 1994.

[44] B. Schölkopf, A. Smola, and K. R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[45] S. Z. Selim and M. A. Ismail. K-means-type algorithms: a generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):81–87, 1984.

[46] P. H. A. Sneath and R. R. Sokal. Numerical taxonomy. *Nature*, 193:855–860, 1962.

[47] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, pages 525–526. Boston, 2000.

[48] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[49] J. Ward. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

[50] D. Wishart. 256. note: An algorithm for hierarchical classifications. *Biometrics*, 25(1):165–170, 1969.

[51] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE transactions on neural networks*, 16(3):645–678, 2005.

[52] K.Y. Yeung, C. Fraley, A. Murua, A.E. Raftery, and W.L. Ruzzo. Model-based clustering and data transformations for gene expression data. *Bioinformatics*, 17(10):977–987, 2001.

[53] C.T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, 100(1):68–86, 1971.