1. **Initialization**: Select 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.

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

   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 is 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, 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 predefined parameter K to obtain a clustering solution and they are often *nondeterministic* 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 sets 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 predefined 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 criterion* (minimum variance). Subsequently, we will also discuss some of the popular divisive clustering methods.

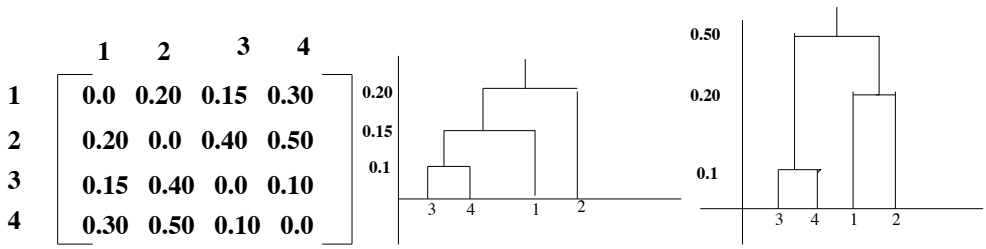
4.3.1 Agglomerative Clustering

The basic steps involved in an agglomerative hierarchical clustering algorithm are the following. First, 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 sets 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 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 neighbor) members. This method intuitively gives more importance to the regions where clusters are closest, 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 behavior, single linkage is capable of effectively clustering nonelliptical, 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 nonlocal in behavior and generally obtains compact shaped clusters. However, similar to single link clustering, this



(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.

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.

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, first 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.

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

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:

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

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 SSE criterion for the clustering obtained by merging them into $C_a \cup C_b$. The Ward's criterion is defined as follows:

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

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 pairs 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 have 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 specify just the new dissimilarity

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 γ = -0.5
Complete Link	$\alpha_i = 0.5$; $\beta = 0$; and $\gamma = 0.5$
GAAC	$\alpha_i = \frac{\lvert i\rvert}{\lvert i\rvert+\lvert j\rvert}$; $\beta = 0$; and $\gamma = 0$
Centroid	$\alpha_i = \frac{\lvert i\rvert}{\lvert i\rvert+\lvert j\rvert}$; $\beta = -\frac{\lvert i\rvert\lvert j\rvert}{(\lvert i\rvert+\lvert j\rvert)^2}$; and $\gamma = 0$
Ward's	$\alpha_i = \frac{\lvert i\rvert+\lvert k\rvert}{\lvert i\rvert+\lvert j\rvert+\lvert k\rvert}$; $\beta = -\frac{\lvert k\rvert}{\lvert i\rvert+\lvert j\rvert+\lvert k\rvert}$; and $\gamma = 0$

between the cluster and all other points. The formula is given as follows:

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

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

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.