Lecture 18

Agglomerative Clustering

Let's start with an overview of hierarchical clustering in general, and then we'll look at agglomerative clustering specifically.

Hierarchical clustering is a method of cluster analysis that seeks to build a hierarchy of clusters. It is an unsupervised learning technique used primarily for data exploration. Unlike methods like K-means clustering that require you to specify the number of clusters upfront, hierarchical clustering can create a dendrogram, a tree-like structure, to visualize the data at various levels of granularity.

How It Works
Hierarchical clustering can be divided into two main types:

1. **Agglomerative Clustering (Bottom-Up Approach)**:
   - *Start with Individual Points:* Each data point is treated as its own cluster.
   - *Merge the Closest Clusters:* The algorithm iteratively merges the two closest clusters based on a distance metric (like Euclidean distance).
   - *Continue Until One Cluster Remains:* This merging process continues until all points belong to a single cluster, forming a hierarchical structure.
   - *Output:* The result can be visualized in a dendrogram, where the x-axis represents the data points and the y-axis represents the distance at which clusters are merged.

2. **Divisive Clustering (Top-Down Approach)**:
   - *Start with All Points in One Cluster:* Initially, all data points are considered to belong to a single cluster.
   - *Recursively Split Clusters:* The algorithm then iteratively splits clusters into smaller clusters, usually by choosing the split that minimizes within-cluster variance or maximizes between-cluster variance.
   - *Continue Until Each Point is Its Own Cluster:* This process continues until each data point is isolated in its own cluster.

Key Concepts

- ***Distance Metrics****:* The distance or similarity between clusters is a key element in hierarchical clustering. Common distance metrics include:
  - *Euclidean Distance:* The straight-line distance between two points.
  - *Manhattan Distance:* The sum of the absolute differences between coordinates.
  - *Cosine Similarity:* Measures the cosine of the angle between two vectors.
- ***Linkage Criteria*****:** Determines how the distance between clusters is calculated:
  - *Single Linkage:* Distance between the closest points of two clusters.
  - *Complete Linkage:* Distance between the furthest points of two clusters.
  - *Average Linkage:* Average distance between all pairs of points in the two clusters.
  - *Centroid Linkage:* Distance between the centroids (mean position of all points) of the clusters.
  - *Ward's Method:* Aims to minimize the total variance within clusters by choosing the merge that results in the least increase in within-cluster variance.
- ***Dendrogram:*** A tree-like diagram that records the sequences of merges or splits. The height at which two clusters are merged or split represents the distance or dissimilarity between them.

Strengths and Weaknesses

Strengths:

- No Need to Specify the Number of Clusters: Hierarchical clustering doesn't require pre-specifying the number of clusters, which can be advantageous when the optimal number of clusters is unknown.
- Dendrogram for Visualization: The dendrogram provides a clear visualization of the clustering process, which can be useful for understanding the structure of the data.
- Flexible: Works with various types of data, including categorical, ordinal, and interval.

Weaknesses:

- Computational Complexity: Especially for large datasets, hierarchical clustering can be computationally intensive, as it requires calculating and storing the distance between each pair of points.
- Sensitivity to Noise and Outliers: Because hierarchical clustering is based on distance calculations, outliers can significantly impact the results.
- Inflexibility: Once a merge or split is done, it cannot be undone. This lack of flexibility can lead to suboptimal clustering solutions.

Applications

Hierarchical clustering is often used in exploratory data analysis, bioinformatics (e.g., gene expression data), text mining, and any field where the relationships among data points can be naturally represented as a hierarchy.

Agglomerative clustering is the most common type of hierarchical clustering, where the algorithm starts by treating each data point as a separate cluster and successively merges pairs of clusters until all points belong to a single cluster. The way clusters are merged is governed by the **linkage criteria**.

The Agglomerative Clustering Process

1. **Initialization**:
    - Begin with n clusters, where n is the number of data points. Each point is its own cluster.
2. **Distance Calculation**:
    - Compute the distance between every pair of clusters. This is done using a specified distance metric, such as Euclidean, Manhattan, or cosine distance.
3. **Merging Clusters**:
    - Identify the pair of clusters with the smallest distance between them, based on the linkage criteria. Merge these two clusters into one.
4. **Updating the Distance Matrix**:
    - After merging two clusters, update the distance matrix to reflect the distances between the new cluster and all other clusters.
5. **Repeat**:
    - Steps 3 and 4 are repeated until only one cluster remains, forming a hierarchy of clusters that can be visualized in a dendrogram.
6. **Dendrogram**:
    - A dendrogram is constructed to show the sequence of merges. The height of each merge in the dendrogram represents the distance (dissimilarity) at which the clusters were merged.

Linkage Criteria

The linkage criterion determines how the distance between two clusters is computed. Different linkage criteria can result in very different clustering solutions, so it's important to understand the implications of each.

1. ***Single Linkage (Minimum Linkage)***
   - Definition: The distance between two clusters is defined as the minimum distance between any single point in the first cluster and any single point in the second cluster.
   - Formula:
     $$D_{single}(C_1, C_2) = \min\{d(x_i, x_j) | x_i \in C_1, x_j \in C_2\}$$
   - Pros:
     - Good for finding elongated or irregularly shaped clusters.
   - Cons:
     - Can result in "chaining," where clusters can become long and stringy, potentially leading to poor cluster formation.

2. ***Complete Linkage (Maximum Linkage)***
   - Definition: The distance between two clusters is defined as the maximum distance between any single point in the first cluster and any single point in the second cluster.
   - Formula:
     $$D_{complete}(C_1, C_2) = \max\{d(x_i, x_j) | x_i \in C_1, x_j \in C_2\}$$
   - Pros:
     - Tends to create more compact, spherical clusters.
   - Cons:
     - Sensitive to outliers and tends to break up larger clusters.

3. ***Average Linkage (Mean Linkage)***
   - Definition: The distance between two clusters is defined as the average distance between all pairs of points where one point is from the first cluster and the other is from the second cluster.
   - Formula:
     $$D_{average}(C_1, C_2) = \frac{1}{|C_1| \times |C_2|} \sum_{x_i \in C_1} \sum_{x_j \in C_2} d(x_i, x_j)$$
   - Pros:
     - Balances between single and complete linkage, reducing the likelihood of chaining and handling outliers better.
   - Cons:
     - Can still suffer from some sensitivity to outliers or non-globular clusters.

4. ***Centroid Linkage***
   - Definition: The distance between two clusters is defined as the distance between the centroids (geometric mean) of the two clusters.
   - Formula:
     $$D_{centroid}(C_1, C_2) = d\left(\frac{1}{|C_1|} \sum_{x_i \in C_1} x_i, \frac{1}{|C_2|} \sum_{x_j \in C_2} x_j\right)$$
   - Pros:

        - Computationally efficient and gives a natural extension of clusters by considering their center points.
    - Cons:
        - Can cause inversions in the dendrogram (where clusters may be merged at different levels than their centroids suggest).

5. ***Ward's Method (Variance Minimization)***
    - Definition: The distance between two clusters is the increase in variance when the two clusters are merged. Ward's method seeks to minimize the total within-cluster variance.
    - Formula:

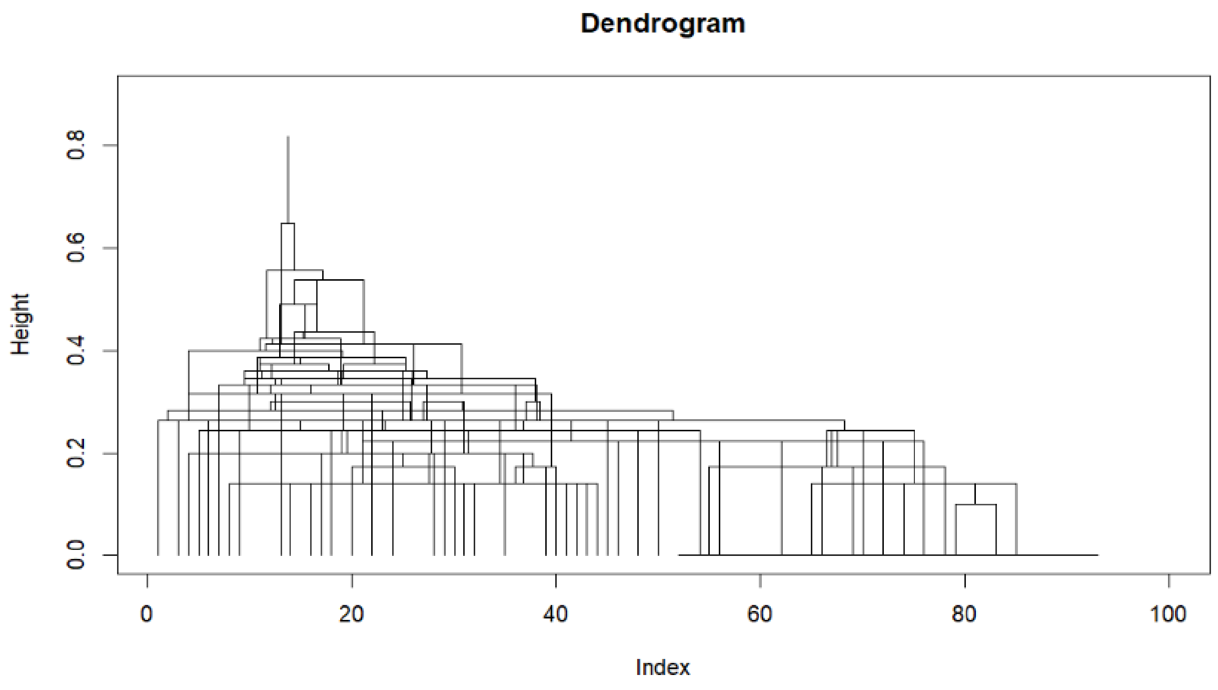
$$D_{Ward}(C_1, C_2) = \sum_{x \in C_1 \cup C_2} d(x, \mu_{C_1 \cup C_2}) - \sum_{x \in C_1} d(x, \mu_{C_1}) - \sum_{x \in C_2} d(x, \mu_{C_2})$$

      where μ_{C_1}, μ_{C_2}, and $\mu_{C_1 \cup C_2}$ are the centroids of clusters C_1, C_2, and their union, respectively.
    - Pros:
        - Often results in more compact and spherical clusters.
        - Effective in balancing the size of clusters.
    - Cons:
        - Computationally intensive.
        - Sensitive to outliers.
6. ***Other Linkage Methods***
    - *Median Linkage:* Similar to centroid linkage, but uses the median instead of the mean to calculate the central point of clusters.
    - *Flexible Linkage:* Allows a customizable blend of other linkages by adjusting a parameter that weights different linkage criteria.

Practical Considerations

- *Choice of Linkage:* The best linkage method depends on the nature of the data and the desired clustering outcome. For example, single linkage is good for discovering non-spherical clusters but can be sensitive to noise, while Ward's method is favored for creating compact, evenly-sized clusters.
- *Computational Complexity:* Agglomerative clustering with different linkage criteria can have different computational costs, with some like Ward's method being more computationally intensive due to variance calculations.
- *Dendrogram Interpretation:* The height of the merge in the dendrogram indicates the distance or dissimilarity at which clusters are combined. This can help in deciding the number of clusters by "cutting" the dendrogram at a certain height.

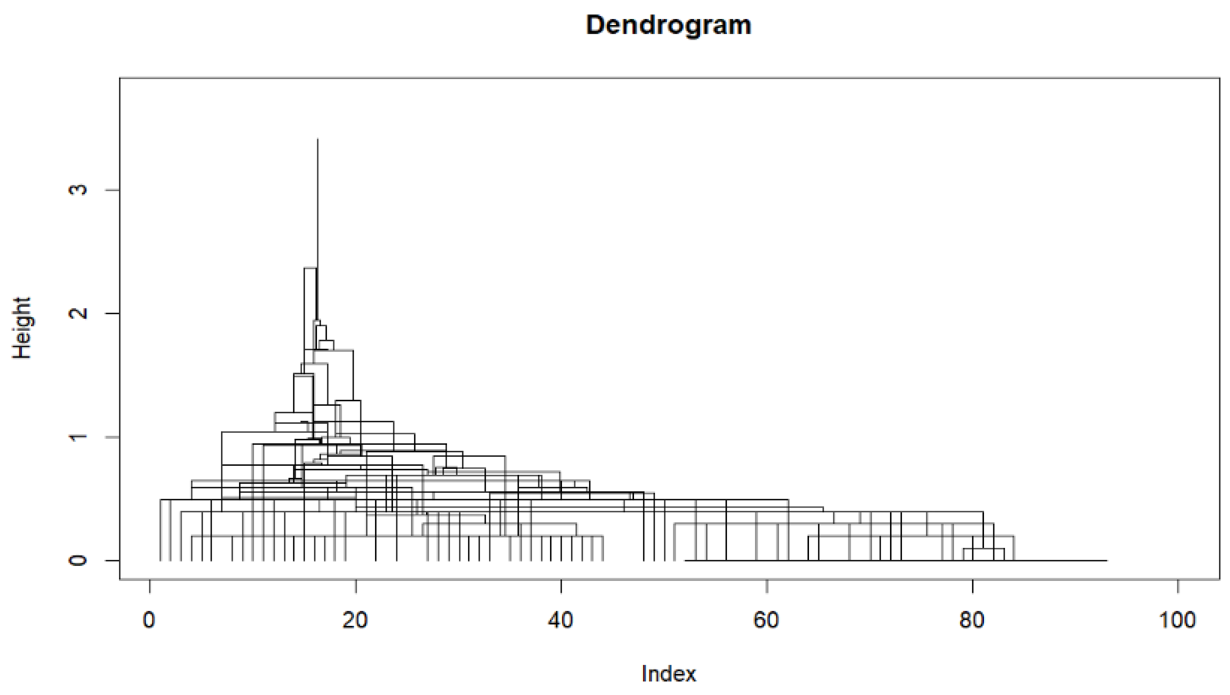
```r
# Step 1: Create Euclidean Distance Matrix
euclidean_distance <- function(data) {
  dist_matrix <- as.matrix(dist(data, method = "euclidean"))
  return(dist_matrix)
}

# Step 2: Perform Single Linkage Agglomerative Clustering
single_linkage_clustering <- function(dist_matrix) {
  n <- nrow(dist_matrix)
  clusters <- list()
  cluster_merges <- list()
  merge_heights <- c()

  for (i in 1:n) {
    clusters[[i]] <- i
  }

  while (length(clusters) > 1) {
    min_dist <- Inf
    closest_clusters <- c(NA, NA)
    for (i in 1:(length(clusters) - 1)) {
      for (j in (i + 1):length(clusters)) {
        cluster_i <- clusters[[i]]
        cluster_j <- clusters[[j]]
        distance <- min(dist_matrix[cluster_i, cluster_j])
        if (distance < min_dist) {
          min_dist <- distance
          closest_clusters <- c(i, j)
        }
      }
    }

    merge_heights <- c(merge_heights, min_dist)
    cluster_merges <- append(cluster_merges, list(closest_clusters))
    clusters[[closest_clusters[1]]] <- c(clusters[[closest_clusters[1]]], clusters[[closest_clusters[2]]])
    clusters[[closest_clusters[2]]] <- NULL
  }

  return(list(cluster_merges = cluster_merges, merge_heights = merge_heights, n = n))
}

# Step 3: Use Iris Data and Remove Setosa
data(iris)
iris_data <- iris[iris$Species != "setosa", -5]

# Step 4: Apply the Clustering
dist_matrix <- euclidean_distance(iris_data)
clustering_result <- single_linkage_clustering(dist_matrix)

# Step 5: Create a Dendrogram
plot_dendrogram <- function(clustering_result) {
  n <- clustering_result$n
  heights <- clustering_result$merge_heights
  merges <- clustering_result$cluster_merges

  # Initialize plot with adjusted height
  plot(0, type = "n", xlim = c(1, n), ylim = c(0, max(heights) * 1.1),
      xlab = "Index", ylab = "Height", main = "Dendrogram")

  cluster_positions <- 1:n
  previous_heights <- rep(0, n)

  for (i in seq_along(merges)) {
    merge <- merges[[i]]
    left <- min(cluster_positions[merge])
    right <- max(cluster_positions[merge])
    mid <- mean(c(left, right))

    # Draw the horizontal line for the current merge height
    segments(left, heights[i], right, heights[i])

    # Draw the vertical lines to the merge height
    segments(left, previous_heights[left], left, heights[i])
    segments(right, previous_heights[right], right, heights[i])

    # Update positions and heights
    cluster_positions[merge] <- mid
    previous_heights[mid] <- heights[i]
  }
}

# Step 6: Plot the Dendrogram
plot_dendrogram(clustering_result)
```



To define the number of final clusters, we'd cut the tree at some height with the desired number of clusters. For example, if we wanted three clusters, we could cut the tree somewhere between 0.5 and 0.55 (we might have to experiment with the exact value).

Let's look at an example where we use Manhattan distance and average linkage.

```
# Step 1: Create Manhattan Distance Matrix
manhattan_distance <- function(data) {
  dist_matrix <- as.matrix(dist(data, method = "manhattan"))
  return(dist_matrix)
}

# Step 2: Perform Average Linkage Agglomerative Clustering
average_linkage_clustering <- function(dist_matrix) {
  n <- nrow(dist_matrix)
  clusters <- list()
  cluster_merges <- list()
  merge_heights <- c()

  for (i in 1:n) {
    clusters[[i]] <- i
  }

  while (length(clusters) > 1) {
    min_avg_dist <- Inf
    closest_clusters <- c(NA, NA)
    for (i in 1:(length(clusters) - 1)) {
      for (j in (i + 1):length(clusters)) {
        cluster_i <- clusters[[i]]
        cluster_j <- clusters[[j]]

        # Calculate average linkage distance
        distance <- mean(dist_matrix[cluster_i, cluster_j])

        if (distance < min_avg_dist) {
          min_avg_dist <- distance
          closest_clusters <- c(i, j)
        }
      }
    }

    merge_heights <- c(merge_heights, min_avg_dist)
    cluster_merges <- append(cluster_merges, list(closest_clusters))
    clusters[[closest_clusters[1]]] <- c(clusters[[closest_clusters[1]]], clusters[[closest_clusters[2]]])
    clusters[[closest_clusters[2]]] <- NULL
  }

  return(list(cluster_merges = cluster_merges, merge_heights = merge_heights, n = n))
}

# Step 3: Use Iris Data and Remove Setosa
data(iris)
iris_data <- iris[iris$Species != "setosa", -5]

# Step 4: Apply the Clustering
dist_matrix <- manhattan_distance(iris_data)
clustering_result <- average_linkage_clustering(dist_matrix)

# Step 5: Create a Dendrogram
plot_dendrogram <- function(clustering_result) {
  n <- clustering_result$n
  heights <- clustering_result$merge_heights
  merges <- clustering_result$cluster_merges

  # Initialize plot with adjusted height
  plot(0, type = "n", xlim = c(1, n), ylim = c(0, max(heights) * 1.1),
       xlab = "Index", ylab = "Height", main = "Dendrogram")

  cluster_positions <- 1:n
  previous_heights <- rep(0, n)

  for (i in seq_along(merges)) {
    merge <- merges[[i]]
    left <- min(cluster_positions[merge])
    right <- max(cluster_positions[merge])
    mid <- mean(c(left, right))

    # Draw the horizontal line for the current merge height
    segments(left, heights[i], right, heights[i])

    # Draw the vertical lines to the merge height
    segments(left, previous_heights[left], left, heights[i])
    segments(right, previous_heights[right], right, heights[i])

    # Update positions and heights
    cluster_positions[merge] <- mid
    previous_heights[mid] <- heights[i]
  }
}

# Step 6: Plot the Dendrogram
plot_dendrogram(clustering_result)
```



Let’s look through examples of the other linkage methods in case we want to apply those (we’ve already looked at other distance methods).

```
complete_linkage <- function(cluster_i, cluster_j, dist_matrix) {
  max_dist <- max(dist_matrix[cluster_i, cluster_j])
  return(max_dist)
}

centroid_linkage <- function(cluster_i, cluster_j, data) {
  centroid_i <- colMeans(data[cluster_i, ])
  centroid_j <- colMeans(data[cluster_j, ])
  centroid_dist <- sqrt(sum((centroid_i - centroid_j)^2))
  return(centroid_dist)
}

wards_linkage <- function(cluster_i, cluster_j, data) {
  combined_cluster <- rbind(data[cluster_i, , drop = FALSE], data[cluster_j, , drop = FALSE])
  total_within_ssq <- sum((data[cluster_i, , drop = FALSE] - colMeans(data[cluster_i, , drop =
FALSE]))^2) +
                    sum((data[cluster_j, , drop = FALSE] - colMeans(data[cluster_j, , drop = FALSE]))^2)
  combined_within_ssq <- sum((combined_cluster - colMeans(combined_cluster))^2)
  increase_in_ssq <- combined_within_ssq - total_within_ssq
  return(increase_in_ssq)
}

median_linkage <- function(cluster_i, cluster_j, data) {
  median_i <- apply(data[cluster_i, , drop = FALSE], 2, median)
  median_j <- apply(data[cluster_j, , drop = FALSE], 2, median)
  median_dist <- sqrt(sum((median_i - median_j)^2))
  return(median_dist)
}
```

The next code segment is an example of how to use these linkages in a clustering loop (compare this section to the full algorithm (it won't run on its own).

```
# Example linkage usage in clustering loop
for (i in 1:(length(clusters) - 1)) {
  for (j in (i + 1):length(clusters)) {
    cluster_i <- clusters[[i]]
    cluster_j <- clusters[[j]]

    # Use the desired linkage method here
    distance <- complete_linkage(cluster_i, cluster_j, dist_matrix)  # Replace with desired method

    if (distance < min_dist) {
      min_dist <- distance
      closest_clusters <- c(i, j)
    }
  }
}
```

You can use this segment, just swap out the linkage method of your choice inside the main algorithm.

Resources:

1. https://www.datanovia.com/en/lessons/agglomerative-hierarchical-clustering/
2. https://www.geeksforgeeks.org/hierarchical-clustering/#
3. https://builtin.com/machine-learning/agglomerative-clustering
4. https://www.geeksforgeeks.org/agglomerative-methods-in-machine-learning/
5. https://www.xlstat.com/en/solutions/features/agglomerative-hierarchical-clustering-ahc