CSCE 970: Pattern Recognition	Spring Semester, 1999

Topic Lecture 9: Hierarchical Clustering Algorithms

Lecturer: Stephen Scott *Scribe: Ingyu Lee*

1 Introduction

In this lecture, we will discuss Hierarchical Clustering Algorithms. Hierarchical Clustering Algorithms is quite different from other clustering algorithms because this kind of algorithms produces a hierarchy of clustering after running instead of producing a single clustering. This kind of algorithms has been used in areas that utilize hierachical group structure, such as biology, archelogy but also is useful for computer science and engineering. There are two types in hierarchical algorithms, the *agglomerative* and the *divisive* hierarchical algorithms. The agglomerative algorithms start from N clusters each containing a single element and repeatedly merge two clusters into one and ends up with a clustering which contains a single set. The divisive algorithms starts with single set clustering and repeatedly divide one cluster into two. This continues until the final clustering which contains N sets. Obviously the divisive algorithms are more computationally expensive. We will focus on the agglomerative algorithms.

2 Review of Definition

- hard(crisp) clustering is defined as $C_i \neq \emptyset, i = 1, \ldots, m$
  $[\cup_{i=1}^{m} C_i] = X$ $C_i \cap C_j = \emptyset, i \neq j,\ i, j \in \{1, \ldots, m\}$ Feature vectors in C_i **more similar** to others in C_i than those in $C_j, j \neq i$
- Divide straight line segment L into N equal parts of length l. Each of the resulting parts is still a strainght segment and its length has been scaled down by a factor $m = frac1L = frac1N$ Magification of any of these parts by the same factor will reproduce the original line segment. These structure as *self-similar.*

3 Generalized Agglomerative Scheme(GAS)

- Define $C_t = \{C_1, \ldots, C_{m_t}\}$ as a **level-t clustering** of X = $\{\mathbf{x_1}, \ldots, \mathbf{x_N}\}$, where C_t meets definition of hard (crisp) clustering.
- A clustering C_t containing k clusters is said to be **nested** in the clustering $C_{t'}$ which contains r (r < k) and we write $C_t \sqsubset C_{t'}$ if each cluster in C_t is a subset of a cluster in $C_{t'}$ and at least one cluster in C_t is a proper subset of some cluster in $C_{t'}$.
  For example, the clustering $C_1 = \{\{\mathbf{x_1}, \mathbf{x_3}\}, \{\mathbf{x_4}\}, \{\mathbf{x_2}, \mathbf{x_5}\}\}$ is nested in $C_2 = \{\{\mathbf{x_1}, \mathbf{x_3}, \mathbf{x_4}\}, \{\mathbf{x_2}, \mathbf{x_5}\}\}$ and we write $C_1 \sqsubset C_2$. However, C_1 is not nested in $C_3 = \{\{\mathbf{x_1}, \mathbf{x_4}\}, \{\mathbf{x_3}\}, \{\mathbf{x_2}, \mathbf{x_5}\}\}$ because the elements $\{\mathbf{x_1}, \mathbf{x_3}\}$ and $\{\mathbf{x_2}, \mathbf{x_5}\}$ are not subset of elements of C_3

The agglomerative algorithms start with C_0 which has N clusters such as $C_0 = \{\{\mathbf{x_1}\}, \ldots, \{\mathbf{x_N}\}\}$ and at each step t merge two clusters into one, yielding C_{t+1} such that $|C_{t+1}| = |C_t| - 1$ and $C_t \sqsubset C_{t+1}$. This procedure continues until the final clustering, C_{N-1} is obtained. At final step

(step N-1), the resulting clusterings is $C_0 \sqsubset C_1 \sqsubset \cdots \sqsubset C_{N-1}$ where $C_0 = \{\{\mathbf{x_1}\}, \ldots, \{\mathbf{x_N}\}\}$ and $C_{N-1} = \{\{\mathbf{x_1}, \ldots, \mathbf{x_N}\}\}$

The divisive algorithms follow the inverse path. The initial clustering C_0 consists of a single set such as $C_0 = \{\{\mathbf{x_1}, \ldots, \mathbf{x_N}\}\}$. At each step t split a cluster into two, yielding C_{t+1} such that $|C_{t+1}| = |C_t| + 1$ and $C_{t+1} \sqsubset C_t$. This procedure continues until the final clustering C_{N-1} is obtained. At final step (step N-1) the resulting clusterings we have :
$C_{N-1} = \{\{\mathbf{x_1}\}, \ldots, \{\mathbf{x_N}\}\} \sqsubset \cdots \sqsubset \mathbf{C_0} = \{\{\mathbf{x_1}, \ldots, \mathbf{x_N}\}\}$

Let $d(C_i, C_j)$ be a function defined for all possible pairs of clusters of X which measures the proximity between C_i and C_j. Let t denote the current level of hierarchy. Then the GAS algorithm as follows

Generalized Agglomerative Scheme (GAS)

1. Choose the initial clustering as $C_0 = \{\{\mathbf{x_1}\}, \ldots, \{\mathbf{x_N}\}\}$, and set $t = 0$
2. For $t = 1$ to $N - 1$
   - Among all possible pairs of clusters (C_r, C_s)
     Find the closest pair of clusters: $(C_i, C_j) = \operatorname{argmin}_{C_s, C_r \in C_{t-1}, r \neq s}\{d(C_s, C_r)\}$
   - Define $C_t = (C_{t-1} - \{C_i, C_j\}) \cup \{\{C_i \cup C_j\}\}$ and update representatives if necessary

This schemes creates a hierarchy of N clusterings, that is $C_t \sqsubset C_s$ for $t < s, s = 1, \ldots, N-1$. The disadvantage of this property is that there is no way to recover from a poor clustering that may have occurred in an earlier level of the hierarchy. If similarity measure is used, then arg max must be used instead of arg min.

At each level t, there are N - t clusters. Thus, in order to determine the pair of clusters that is going to be merged at the t + 1 level, $\binom{N-1}{2} \equiv \frac{(N-t)(N-t-1)}{2}$ pairs of clusters have to be considered. Thus, the total number of pairs that have to be examined is $\frac{(N-1)N(N+1)}{6}$ that is $O(N^3)$

4 Implementing GAS

There are two main categories of agglomerative algorithms. First category are based on matrix theory and the second category are based on graph theory. We will focus on matrix theory.

- The pattern matrix $D(X)$ is the $N \times l$ matrix, whose ith row is the ith vector of X.
- The similarity (dissimilarity) matrix, P(X), is an $N \times N$ matrix whose (i,j) element equals the similarity $s(x_i, x_j)$(dissimilarity $d(x_i, x_j)$) between vectors x_i and X_j.
- An $(N-t) \times (N-t)$ proximity matrix P_t gives the proximity between all pairs of clusters at level t. It's elements equals the similarity $s(x_i, y_j)$ or dissimilarity $s(x_i, y_j)$ between vectors x_i and x_j
- A dendrogram is an effective means of representing the sequence of clusterings produced by an agglomerative algorithm. Cutting the dendrogram at a specific level results in a clustering. A proximity dendogram is a tree that indicates hierarchy of clusterings, including the proximity between two clusters when they are merged.

4.1 Example of GAS

For example $\mathbf{x_1} = [\mathbf{1}, \mathbf{1}]^{\mathbf{T}}$, $\mathbf{x_2} = [\mathbf{2}, \mathbf{1}]^{\mathbf{T}}$, $\mathbf{x_3} = [\mathbf{5}, \mathbf{4}]^{\mathbf{T}}$, $\mathbf{x_4} = [\mathbf{6}, \mathbf{5}]^{\mathbf{T}}$, $\mathbf{x_5} = [\mathbf{6.5}, \mathbf{6}]^{\mathbf{T}}$, DM = Euclidian/$\alpha^{ss}_{min}$, SM=Tanimoto/$\alpha^{ss}_{max}$.

Then, the pattern matrix D(x), similarity matrix P_0^{DM} and dissimilarity matrix P_0^{SM} has the following value.

$$D(x) = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 5 & 4 \\ 6 & 5 \\ 6.5 & 6 \end{bmatrix}$$

$$P_0^{DM} = \begin{bmatrix} 0 & 1 & 5 & 6.4 & 7.4 \\ 1 & 0 & 4.2 & 5.7 & 6.7 \\ 5 & 4.2 & 0 & 1.4 & 2.5 \\ 6.4 & 5.7 & 1.4 & 0 & 1.1 \\ 7.4 & 6.7 & 2.5 & 1.1 & 0 \end{bmatrix}$$

$$P_0^{SM} = \begin{bmatrix} 1 & 0.75 & 0.26 & 0.21 & 0.18 \\ 0.75 & 1 & 0.44 & 0.35 & 0.20 \\ 0.26 & 0.44 & 1 & 0.96 & 0.90 \\ 0.21 & 0.35 & 0.96 & 1 & 0.98 \\ 0.18 & 0.20 & 0.90 & 0.98 & 1 \end{bmatrix}$$

Figure 1 shows how it different using SM or DM. (a) is the proximity (similarity) dendrogram for X using P_0^{SM} and (b) is the proximity (dissimilarity) dendrogram for X using P_0^{DM} from above example. If we cut at 0.8 for (a) and 2 for (b), then the result is quite different. The result using (a) is $\{\{x_1\}, \{x_2\}, \{x_3, x_4, x_5\}\}$ and the result using (b) is $\{\{x_1, x_2\}, \{x_3, x_4, x_5\}\}$.

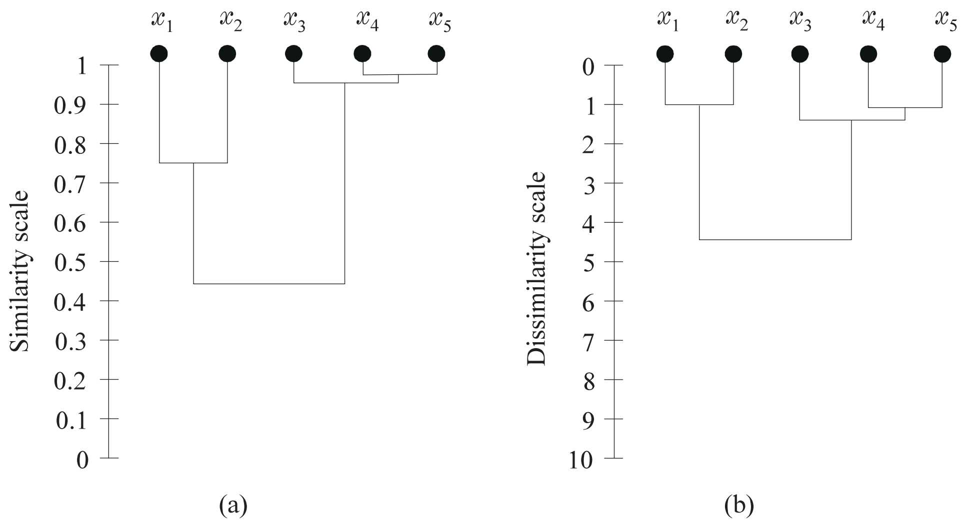

Figure 1: (a) similarity dendrogram (b) dissimilarity dendrogram

4.2 Matrix Updating Algorithmic Scheme(MUAS)

The input in these schemes is the $N \times N$ dissimilarity matrix, P_0. At each level t, when two clusters are merged into one, the size of the dissimilarity matrix P_t becomes $(N-t) \times (N-t)$. Delete the two rows and columns that correspond to the merged clusters and add a new row and a new column that contains the distances between the newly formed cluster and the old clusters at each step. Following is the algorithm.

1. Initialize $C_0 = \{\{\mathbf{x_1}\}, \dots, \{\mathbf{x_N}\}\}$, t = 0, P_0
2. For $t = 1$ to $N-1$
   - Find closest pair of clusters by scanning upper diagonal of P_{t-1} for min(DM) or max(SM), yielding C_i and C_j
   - $C_t = (C_{t-1} - \{C_i, C_j\}) \cup \{\{C_i \cup C_j\}\}$
   - Update P_{t-1} to get P_t:
     Delete rows i and j and columns i and j from P_{t-1} and add a row and column for $C_q = C_i \cup C_j$, where the sth entry is
     $\alpha(C_q, C_s) = \min\{\alpha(C_i, C_s), \alpha(C_j, C_s)\}$ OR $\alpha(C_q, C_s) = \max\{\alpha(C_i, C_s), \alpha(C_j, C_s)\}$

If we use $\min\{\alpha(C_i, C_s), \alpha(C_j, C_s)\}$, then it yields single link algorithm for DM and If we use $\max\{alpha(C_i, C_s), \alpha(C_j, C_s)\}$ it yields complete link algorithm for SM.

Below is an example of single-link, squared euclidian Distance. Start with $C_0 = \{\{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}, \mathbf{x_5}\}\}$ At first step, finds $X_1 and X_2$ has the minimum DM 1 by the P_0. So, delete the first and second row, column. Add a new column and row with value of minimum distance. The result is P_1. At the second step, we find X_4 and X_5 has the minimum DM 1.5 by the P_1 and do the same things. Now, we have C_2 and P_2. In this way, we can get C_3 and P_3 and C_4 and P_4.

$$C_0 = \{\{\mathbf{x_1}\}, \{\mathbf{x_2}\}, \{\mathbf{x_3}\}, \{\mathbf{x_4}\}, \{\mathbf{x_5}\}\}$$
$$C_1 = \{\{\mathbf{x_1}, \mathbf{x_2}\}, \{\mathbf{x_3}\}, \{\mathbf{x_4}\}, \{\mathbf{x_5}\}\}$$
$$C_2 = \{\{\mathbf{x_1}, \mathbf{x_2}\}, \{\mathbf{x_3}\}, \{\mathbf{x_4}, \mathbf{x_5}\}\}$$
$$C_3 = \{\{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}\}, \{\mathbf{x_4}, \mathbf{x_5}\}\}$$
$$C_4 = \{\{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}, \mathbf{x_5}\}\}$$

$$P_0 = \begin{bmatrix} 0 & 1 & 2 & 26 & 37 \\ 1 & 0 & 3 & 25 & 36 \\ 2 & 3 & 0 & 16 & 25 \\ 26 & 25 & 16 & 0 & 15 \\ 37 & 36 & 25 & 1.5 & 0 \end{bmatrix}$$

$$P_1 = \begin{bmatrix} 0 & 2 & 25 & 36 \\ 2 & 0 & 16 & 25 \\ 25 & 16 & 0 & 1.5 \\ 36 & 25 & 1.5 & 0 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 0 & 2 & 25 \\ 2 & 0 & 16 \\ 25 & 16 & 0 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 0 & 16 \\ 16 & 0 \end{bmatrix}$$

$$P_4 = [0]$$

4.3 Single Versus Complete-Link Algorithms

Single-link algorithm tends to favor elongated clusters since only require a single member of one cluster to be near a single member of another. Complete-link algorithm tends to favor compact clusters since require all members of one cluster to be near all members of another. Single-link algorithm produces same dendogram regardless of how ties are broken in proximity matrix but complete-link and others do not possess this property.

Below shows the difference between single-link and complete-link algorithms. (a) is a feature data sets. Using single-link algorithm, the results will be (b) and using complete-link algorithm, the results will be (c). When we cut between 1.5 and 2, then the single-link algorithm produces $\{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$ and $\{x_8\}, \{x_9\}, \{x_10\}, \{x_11\}$. The result is a one elongated cluster and four individual points. However, if we try on complete-link algorithm, it produces $\{x_1, x_2\}, \{x_3, x_4\}, \{x_5, x_6\}, \{x_7\}, \{x_8\}, \{x_9\}, \{x_{10}\}, \{x_{11}\}$. It shows quite diferent results.

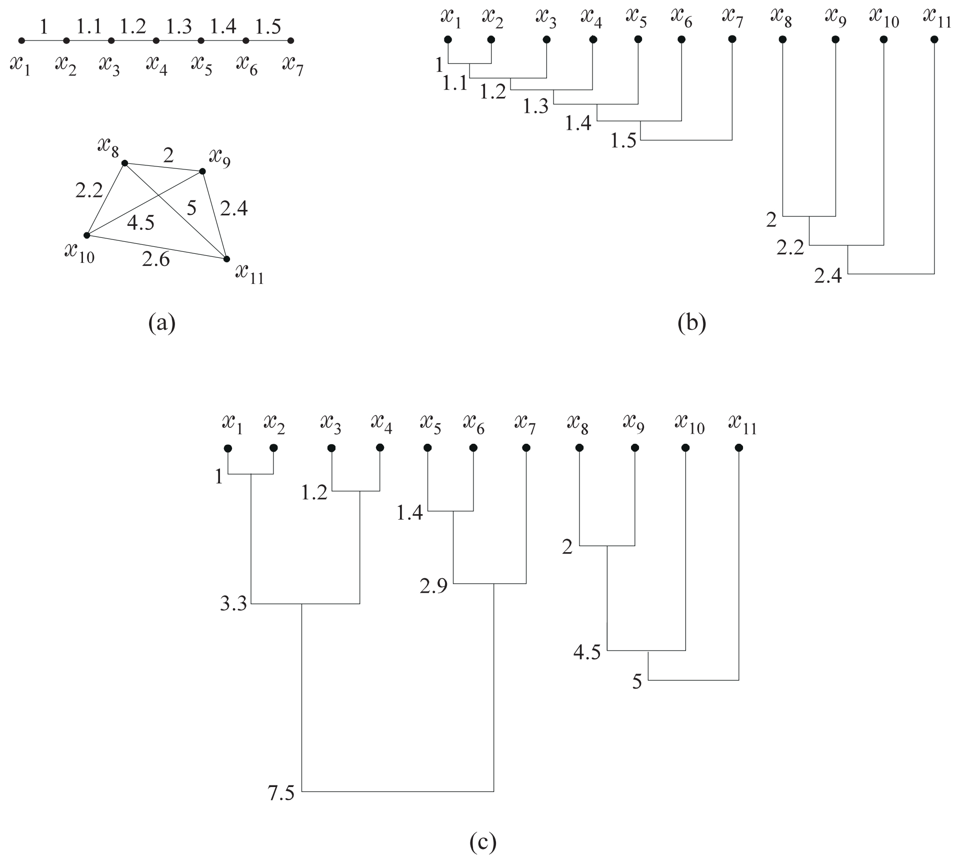

Figure 2: (a) data set X (b) Using single link (c) Using complete link

5 Where to Cut the Dendogram

There are two methods to cut the dendogram. First, using clusters with long lifetimes. Lifetime is the difference in proximity level between creation and absorption. Below shows example of clustering using lifetime. (a) has two major clusters and (b) has one major cluster when we cut using lifetime.

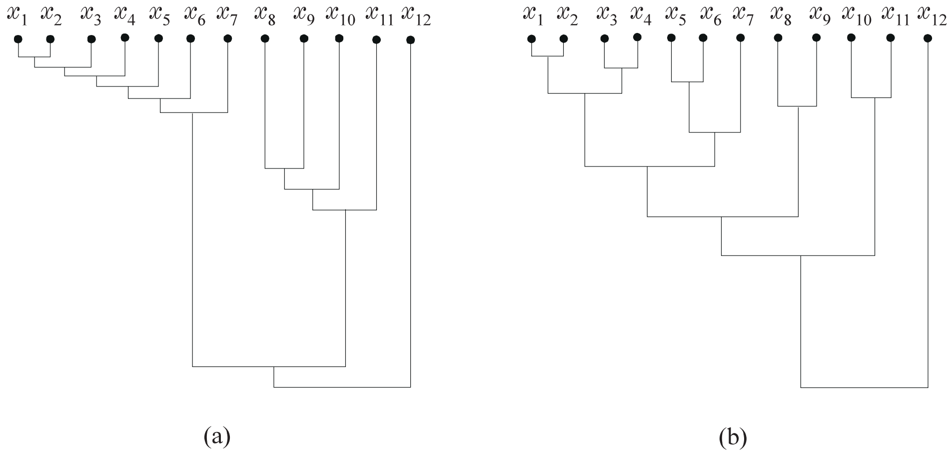

Figure 3: [(a) A dendrogram that suggests that there are two major clusters in the data set (b) A dendrogram indicating that that there is a single major cluster in the data set

Second, using self-similarity. If we define *self-similarity* as $h(C) = max_{\mathbf{X},\mathbf{Y}\in\mathbf{C}}\{d(X,Y)\}$, and use clustering C_t such that for the first time $\exists C_j \in C_{t+1} : h(C_j) > \theta$ or $\alpha^{ss}_{min}(C_i, C_j) > max\{h(C_i), h(C_j)\}, \forall C_i, C_j \in C_t$