CS481: Bioinformatics Algorithms

Can Alkan
EA509
calkan@cs.bilkent.edu.tr

http://www.cs.bilkent.edu.tr/~calkan/teaching/cs481/
CLUSTERING
Applications of Clustering

- Viewing and analyzing vast amounts of biological data as a whole set can be infeasible.

- It is easier to interpret the data if they are partitioned into clusters combining similar data points.
Homogeneity and Separation Principles

- **Homogeneity:** Elements within a cluster are close to each other
- **Separation:** Elements in different clusters are further apart from each other
- …clustering is not an easy task!

Given these points a clustering algorithm might make two distinct clusters as follows
Bad Clustering

This clustering violates both Homogeneity and Separation principles.

- Close distances from points in separate clusters
- Far distances from points in the same cluster
Good Clustering

This clustering satisfies both Homogeneity and Separation principles.
Clustering Techniques

- **Agglomerative**: Start with every element in its own cluster, and iteratively join clusters together.

- **Divisive**: Start with one cluster and iteratively divide it into smaller clusters.

- **Hierarchical**: Organize elements into a tree, leaves represent data points and the length of the paths between leaves represents the distances between data points. Similar data points lie within the same subtrees.
Hierarchical Clustering
Hierarchical Clustering: Example
Hierarchical Clustering: Example
Hierarchical Clustering: Example
Hierarchical Clustering: Example
Hierarchical Clustering: Example
Hierarchical Clustering Algorithm

1. **Hierarchical Clustering** \((d, n)\)
2. Form \(n\) clusters each with one element
3. Construct a graph \(T\) by assigning one vertex to each cluster
4. **while** there is more than one cluster
5. Find the two closest clusters \(C_1\) and \(C_2\)
6. Merge \(C_1\) and \(C_2\) into new cluster \(C\) with \(|C_1| + |C_2|\) elements
7. **Compute distance from** \(C\) **to all other clusters**
8. Add a new vertex \(C\) to \(T\) and connect to vertices \(C_1\) and \(C_2\)
9. Remove rows and columns of \(d\) corresponding to \(C_1\) and \(C_2\)
10. Add a row and column to \(d\) corresponding to the new cluster \(C\)
11. return \(T\)

The algorithm takes a \(nxn\) distance matrix \(d\) of pairwise distances between points as an input.
Hierarchical Clustering Algorithm

1. Hierarchical Clustering \((d, n)\)
2. Form \(n\) clusters each with one element
3. Construct a graph \(T\) by assigning one vertex to each cluster
4. while there is more than one cluster
   5. Find the two closest clusters \(C_1\) and \(C_2\)
   6. Merge \(C_1\) and \(C_2\) into new cluster \(C\) with \(|C_1| + |C_2|\) elements
   7. Compute distance from \(C\) to all other clusters
   8. Add a new vertex \(C\) to \(T\) and connect to vertices \(C_1\) and \(C_2\)
   9. Remove rows and columns of \(d\) corresponding to \(C_1\) and \(C_2\)
   10. Add a row and column to \(d\) corresponding to the new cluster \(C\)
11. return \(T\)

Different ways to define distances between clusters may lead to different clusterings
Hierarchical Clustering: Recomputing Distances

- \[ d_{\text{min}}(C, C^*) = \min d(x,y) \]
  for all elements \( x \) in \( C \) and \( y \) in \( C^* \)

- Distance between two clusters is the **smallest** distance between any pair of their elements

- \[ d_{\text{avg}}(C, C^*) = \frac{1}{|C^*||C|} \sum d(x,y) \]
  for all elements \( x \) in \( C \) and \( y \) in \( C^* \)

- Distance between two clusters is the **average** distance between all pairs of their elements
**Squared Error Distortion**

- Given a data point $v$ and a set of points $X$, define the **distance** from $v$ to $X$

  $$d(v, X)$$

  as the (Euclidean) distance from $v$ to the **closest** point from $X$.

- Given a set of $n$ data points $V = \{v_1 ... v_n\}$ and a set of $k$ points $X$, define the **Squared Error Distortion**

  $$d(V, X) = \frac{\sum d(v_i, X)^2}{n} \quad 1 \leq i \leq n$$
K-Means Clustering Problem: Formulation

- **Input**: A set, \( V \), consisting of \( n \) points and a parameter \( k \)

- **Output**: A set \( X \) consisting of \( k \) points (cluster centers) that minimizes the squared error distortion \( d(V,X) \) over all possible choices of \( X \)
1-Means Clustering Problem: an Easy Case

- **Input**: A set, $V$, consisting of $n$ points

- **Output**: A single point $x$ (*cluster center*) that minimizes the squared error distortion $d(V,x)$ over all possible choices of $x$
1-Means Clustering Problem: an Easy Case

- **Input**: A set, $V$, consisting of $n$ points

- **Output**: A single point $x$ (cluster center) that minimizes the squared error distortion $d(V, x)$ over all possible choices of $x$

1-Means Clustering problem is easy.

However, it becomes very difficult (NP-complete) for more than one center.

An efficient *heuristic* method for K-Means clustering is the Lloyd algorithm.
K-Means Clustering: Lloyd Algorithm

1. **Lloyd Algorithm**
2. Arbitrarily assign the $k$ cluster centers
3. **while** the cluster centers keep changing
4. Assign each data point to the cluster $C_i$ corresponding to the closest cluster representative (center) $(1 \leq i \leq k)$
5. After the assignment of all data points, compute new cluster representatives according to the center of gravity of each cluster, that is, the new cluster representative is
   \[ \sum v / |C| \text{ for all } v \text{ in } C \] for every cluster $C$

*This may lead to merely a locally optimal clustering.*
Conservative K-Means Algorithm

- Lloyd algorithm is fast but in each iteration it moves many data points, not necessarily causing better convergence.
- A more conservative method would be to move one point at a time only if it improves the overall clustering cost.
  - The smaller the clustering cost of a partition of data points is the better that clustering is.
  - Different methods (e.g., the squared error distortion) can be used to measure this clustering cost.
K-Means Greedy Algorithm

1. **ProgressiveGreedyK-Means($k$)**
2. Select an arbitrary partition $P$ into $k$ clusters
3. **while** forever
4. \[ \text{bestChange} \leftarrow 0 \]
5. for every cluster $C$
6. for every element $i$ not in $C$
7. if moving $i$ to cluster $C$ reduces its clustering cost
8. \[ \text{if } (\text{cost}(P) - \text{cost}(P_{i \rightarrow C}) > \text{bestChange} \]
9. \[ \text{bestChange} \leftarrow \text{cost}(P) - \text{cost}(P_{i \rightarrow C}) \]
10. \[ i^* \leftarrow i \]
11. \[ C^* \leftarrow C \]
12. if $\text{bestChange} > 0$
13. Change partition $P$ by moving $i^*$ to $C^*$
14. else
15. return $P$
CLUSTERING USING GRAPHS
Clique Graphs

- A **clique** is a graph with every vertex connected to every other vertex.
- A **clique graph** is a graph where each connected component is a clique.
A graph can be transformed into a clique graph by adding or removing edges.
Corrupted Cliques Problem

**Input:** A graph $G$

**Output:** The smallest number of additions and removals of edges that will transform $G$ into a clique graph
Distance Graphs

- Turn the distance matrix into a distance graph
  - Genes are represented as vertices in the graph
  - Choose a distance threshold $\theta$
  - If the distance between two vertices is below $\theta$, draw an edge between them
  - The resulting graph may contain cliques
  - These cliques represent clusters of closely located data points
Transforming Distance Graph into Clique Graph

The distance graph (threshold $\theta=7$) is transformed into a clique graph after removing the two highlighted edges.

After transforming the distance graph into the clique graph, the dataset is partitioned into three clusters.

![Figure 10.6](image)

The distance graph (b) for $\theta=7$ is not quite a clique graph. However, it can be transformed into a clique graph (c) by removing edges $(g_1, g_{10})$ and $(g_1, g_9)$. 

(a) Distance matrix, $d$ (distances shorter than 7 are shown in bold).
Corrupted Cliques problem is NP-Hard, some heuristics exist to approximately solve it:

**CAST** (Cluster Affinity Search Technique): a practical and fast algorithm:

- **CAST** is based on the notion of genes close to cluster $C$ or distant from cluster $C$
- Distance between gene $i$ and cluster $C$:

$$d(i,C) = \text{average distance between gene } i \text{ and all genes in } C$$

Gene $i$ is **close** to cluster $C$ if $d(i,C) < \theta$ and **distant** otherwise
CAST Algorithm

1. CAST(S, G, θ)
2. \( P \leftarrow \emptyset \)
3. while \( S \neq \emptyset \)
4. \( V \leftarrow \) vertex of maximal degree in the distance graph \( G \)
5. \( C \leftarrow \{ v \} \)
6. while a close gene \( i \) not in \( C \) or distant gene \( i \) in \( C \) exists
7. Find the nearest close gene \( i \) not in \( C \) and add it to \( C \)
8. Remove the farthest distant gene \( i \) in \( C \)
9. Add cluster \( C \) to partition \( P \)
10. \( S \leftarrow S \setminus C \)
11. Remove vertices of cluster \( C \) from the distance graph \( G \)
12. return \( P \)

\( S \) – set of elements, \( G \) – distance graph, \( \theta \) – distance threshold
CAST Algorithm

\[ \Theta = 7 \]
\[ P = \emptyset \]
\[ S = \{ g_1, \ldots, g_{10} \} \]
\[ \text{degree}(g_{10}) = 4 \]

\[ C_1 = \{ g_{10} \} \]
\[ C_1 = \{ g_2, g_{10} \} \]

\[ d(g_1, C_1) = \frac{7 + 8.1}{2} = 7.55 \]
\[ d(g_4, C_1) = \frac{0.9 + 1.1}{2} = 1 \]
\[ d(g_9, C_1) = \frac{2 + 1.1}{2} = 1.55 \]

\[ C_1 = \{ g_2, g_4, g_{10} \} \]
\[ d(g_9, C) = \frac{2 + 1.6 + 1}{3} = 1.53 \]
\[ C_1 = \{ g_2, g_4, g_9, g_{10} \} \]
\[ P = \{ C_1 \} \]
CAST Algorithm

$\Theta = 7$

$P = \{C_1\}$

$C_1 = \{g_2, g_4, g_9, g_{10}\}$

$S = \{g_1, g_3, g_5, g_6, g_7, g_8\}$

$\text{degree}(g_1) = 2$

$C_2 = \{g_1\}$

$C_2 = \{g_1, g_6\}$

$d(g_7, C_2) = (5.1+5.6) / 2 = 5.35$

$C_2 = \{g_1, g_6, g_7\}$

$P = \{C_1, C_2\}$
CAST Algorithm

\[ \Theta = 7 \]
\[ P = \{C_1, C_2\} \]
\[ C_1 = \{g_2, g_4, g_9, g_{10}\} \]
\[ C_2 = \{g_1, g_6, g_7\} \]
\[ S = \{g_3, g_5, g_8\} \]
\[ \text{degree}(g_3) = 2 \]

\[ C_3 = \{g_3\} \]
\[ C_3 = \{g_3, g_5\} \]

\[ d(g_8, C_3) = \frac{(1.1+1)}{2} = 1.05 \]
\[ C_3 = \{g_3, g_5, g_8\} \]
\[ P = \{C_1, C_2, C_3\} \]
CAST Algorithm

\[ \Theta = 7 \]
\[ P = \{ C_1, C_2, C_3 \} \]
\[ C_1 = \{ g_2, g_4, g_9, g_{10} \} \]
\[ C_2 = \{ g_1, g_6, g_7 \} \]
\[ C_3 = \{ g_3, g_5, g_8 \} \]
\[ S = \emptyset \]

... done