CS685: Data Mining
Hierarchical Clustering Methods

Arnab Bhattacharya
arnabb@cse.iitk.ac.in

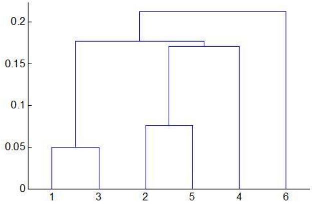
Computer Science and Engineering,
Indian Institute of Technology, Kanpur
http://web.cse.iitk.ac.in/~cs685/

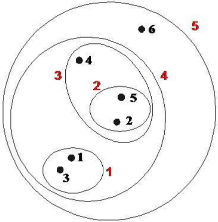
1st semester, 2020-21

Mon 1030-1200 (online)

Hierarchical Clustering Methods

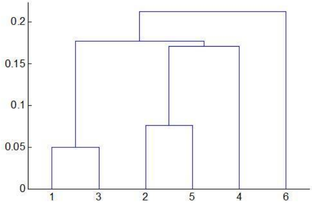
- Arranges the objects in a hierarchy (tree)
- Can be visualized as a dendogram or a nested cluster

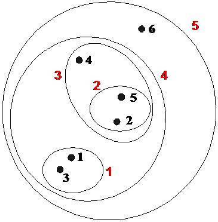




Hierarchical Clustering Methods

- Arranges the objects in a hierarchy (tree)
- Can be visualized as a dendogram or a nested cluster

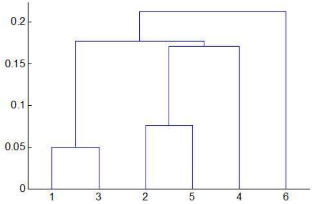


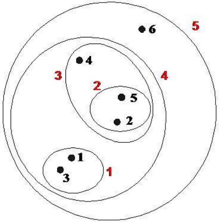


- Can proceed bottom-up (agglomerative) or top-down (divisive)

Hierarchical Clustering Methods

- Arranges the objects in a hierarchy (tree)
- Can be visualized as a dendogram or a nested cluster





- Can proceed bottom-up (agglomerative) or top-down (divisive)
- Requires defining distance between clusters, i.e., sets of objects

$$min(A, B) = \min_{p \in A, q \in B} \{d(p, q)\}$$

$$max(A, B) = \max_{p \in A, q \in B} \{d(p, q)\}$$

$$avg(A, B) = \sum_{p \in A, q \in B} d(p, q)/(|A|.|B|)$$

$$mean(A, B) = d(\mu_A, \mu_B)$$

Agglomerative Hierarchical Method

- AGglomerative NESting (AGNES)

Agglomerative Hierarchical Method

- AGglomerative NESting (AGNES)
- Starts with each object as a cluster
- In each step, selects the two clusters with the least minimum distance and merges them

Divisive Hierarchical Method

- DIvisive ANAlysis (DIANA)

Divisive Hierarchical Method

- DIvisive ANAlysis (DIANA)
- Starts with all the objects in one big cluster
- In each step, selects the cluster with the largest *diameter* and splits it
  - Diameter is the largest pairwise distance between two objects in the cluster

Divisive Hierarchical Method

- DIvisive ANAlysis (DIANA)
- Starts with all the objects in one big cluster
- In each step, selects the cluster with the largest *diameter* and splits it
  - Diameter is the largest pairwise distance between two objects in the cluster
- Object that has the largest average distance to other members of the cluster is chosen
- This object initiates the splinter group
- For an object, average distance to the splinter group and non-splinter group is found
- If splinter group is closer, it is merged with the splinter group

Divisive Hierarchical Method

- DIvisive ANAlysis (DIANA)
- Starts with all the objects in one big cluster
- In each step, selects the cluster with the largest *diameter* and splits it
  - Diameter is the largest pairwise distance between two objects in the cluster
- Object that has the largest average distance to other members of the cluster is chosen
- This object initiates the splinter group
- For an object, average distance to the splinter group and non-splinter group is found
- If splinter group is closer, it is merged with the splinter group
- The splinter group forms a new cluster out of the parent

Divisive Hierarchical Method

- DIvisive ANAlysis (DIANA)
- Starts with all the objects in one big cluster
- In each step, selects the cluster with the largest *diameter* and splits it
  - Diameter is the largest pairwise distance between two objects in the cluster
- Object that has the largest average distance to other members of the cluster is chosen
- This object initiates the splinter group
- For an object, average distance to the splinter group and non-splinter group is found
- If splinter group is closer, it is merged with the splinter group
- The splinter group forms a new cluster out of the parent
- Order dependent

Analysis of Hierarchical Methods

- For n objects, the number of merging or splitting is

Analysis of Hierarchical Methods

- For n objects, the number of merging or splitting is $n - 1$
- If there are m clusters in an agglomeration step, the number of possibilities of merging is

Analysis of Hierarchical Methods

- For n objects, the number of merging or splitting is $n - 1$
- If there are m clusters in an agglomeration step, the number of possibilities of merging is $m(m - 1)/2$
- If there are m clusters in a division step, the number of clusters that can be split is

Analysis of Hierarchical Methods

- For n objects, the number of merging or splitting is $n - 1$
- If there are m clusters in an agglomeration step, the number of possibilities of merging is $m(m-1)/2$
- If there are m clusters in a division step, the number of clusters that can be split is m
- For a cluster with t objects, the number of ways it can be split is

Analysis of Hierarchical Methods

- For n objects, the number of merging or splitting is $n - 1$
- If there are m clusters in an agglomeration step, the number of possibilities of merging is $m(m - 1)/2$
- If there are m clusters in a division step, the number of clusters that can be split is m
- For a cluster with t objects, the number of ways it can be split is $2^{t-1} - 1$
- In general, agglomerative methods are easier than divisive methods
- Hence, there are more agglomerative methods in practice

Distances for Agglomerative Methods

- *Minimum pairwise distance* methods are nearest-neighbor algorithms
  - Also called single linkage algorithms
  - If a distance threshold is chosen, a single link between two clusters needs to be below the threshold for the clusters to be merged
  - Finally forms a spanning tree of objects
  - Can form non-elliptical clusters

Distances for Agglomerative Methods

- *Minimum pairwise distance* methods are nearest-neighbor algorithms
  - Also called single linkage algorithms
  - If a distance threshold is chosen, a single link between two clusters needs to be below the threshold for the clusters to be merged
  - Finally forms a spanning tree of objects
  - Can form non-elliptical clusters
- *Maximum pairwise distance* methods are farthest-neighbor algorithms
  - Also called complete linkage algorithms
  - If a distance threshold is chosen, all links between two clusters needs to be below the threshold for the clusters to be merged
  - Finally forms a clique of objects
  - Favours forming elliptical clusters

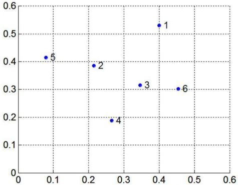
Distances for Agglomerative Methods

- *Minimum pairwise distance* methods are nearest-neighbor algorithms
  - Also called single linkage algorithms
  - If a distance threshold is chosen, a single link between two clusters needs to be below the threshold for the clusters to be merged
  - Finally forms a spanning tree of objects
  - Can form non-elliptical clusters
- *Maximum pairwise distance* methods are farthest-neighbor algorithms
  - Also called complete linkage algorithms
  - If a distance threshold is chosen, all links between two clusters needs to be below the threshold for the clusters to be merged
  - Finally forms a clique of objects
  - Favours forming elliptical clusters
- *Average pairwise distance* is robust to noise and outliers

Distances for Agglomerative Methods

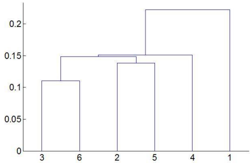
- *Minimum pairwise distance* methods are nearest-neighbor algorithms
  - Also called single linkage algorithms
  - If a distance threshold is chosen, a single link between two clusters needs to be below the threshold for the clusters to be merged
  - Finally forms a spanning tree of objects
  - Can form non-elliptical clusters
- *Maximum pairwise distance* methods are farthest-neighbor algorithms
  - Also called complete linkage algorithms
  - If a distance threshold is chosen, all links between two clusters needs to be below the threshold for the clusters to be merged
  - Finally forms a clique of objects
  - Favours forming elliptical clusters
- *Average pairwise distance* is robust to noise and outliers
- *Centroid-based distances* (such as mean) may exhibit an undesirable property called inversion
  - Distance between clusters merged at a later step may be *less* than those merged earlier

Example

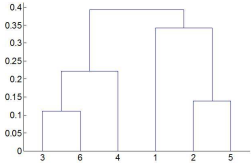


Point	x	y
1	0.40	0.53
2	0.22	0.38
3	0.35	0.32
4	0.26	0.19
5	0.08	0.41
6	0.45	0.30

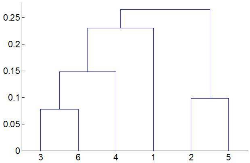
	1	2	3	4	5	6
1	0.00	0.24	0.22	0.37	0.34	0.23
2		0.00	0.15	0.20	0.14	0.25
3			0.00	0.15	0.28	0.11
4				0.00	0.29	0.22
5					0.00	0.39
6						0.00



Minimum



Maximum



Average

Minimum Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	37	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

Minimum Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	37	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	22	37	34
2		0	15	20	14
3,6			0	15	28
4				0	29
5					0

Minimum Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	37	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	22	37	34
2		0	15	20	14
3,6			0	15	28
4				0	29
5					0

$\xrightarrow{(2,5)}$

	1	2,5	3,6	4
1	0	24	22	37
2,5		0	15	20
3,6			0	15
4				0

Minimum Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	37	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	22	37	34
2		0	15	20	14
3,6			0	15	28
4				0	29
5					0

$\xrightarrow{(2,5)}$

	1	2,5	3,6	4
1	0	24	22	37
2,5		0	15	20
3,6			0	15
4				0

$\xrightarrow{((2,5),(3,6))}$

	1	2,5,3,6	4
1	0	22	37
2,5,3,6		0	15
4			0

Minimum Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	37	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	22	37	34
2		0	15	20	14
3,6			0	15	28
4				0	29
5					0

$\xrightarrow{(2,5)}$

	1	2,5	3,6	4
1	0	24	22	37
2,5		0	15	20
3,6			0	15
4				0

$\xrightarrow{((2,5),(3,6))}$

	1	2,5,3,6	4
1	0	22	37
2,5,3,6		0	15
4			0

$\xrightarrow{(((2,5),(3,6)),4)}$

	1	2,5,3,6,4
1	0	22
2,5,3,6,4		0

Minimum Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	37	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	22	37	34
2		0	15	20	14
3,6			0	15	28
4				0	29
5					0

$\xrightarrow{(2,5)}$

	1	2,5	3,6	4
1	0	24	22	37
2,5		0	15	20
3,6			0	15
4				0

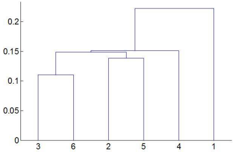
$\xrightarrow{((2,5),(3,6))}$

	1	2,5,3,6	4
1	0	22	37
2,5,3,6		0	15
4			0

$\xrightarrow{(((2,5),(3,6)),4)}$

	1	2,5,3,6,4
1	0	22
2,5,3,6,4		0

$\rightarrow ((((2,5),(3,6)),4),1)$



Maximum Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	37	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

Maximum Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	37	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	23	37	34
2		0	25	20	14
3,6			0	22	39
4				0	29
5					0

Maximum Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	37	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	23	37	34
2		0	25	20	14
3,6			0	22	39
4				0	29
5					0

$\xrightarrow{(2,5)}$

	1	2,5	3,6	4
1	0	34	23	37
2,5		0	39	29
3,6			0	22
4				0

Maximum Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	37	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	23	37	34
2		0	25	20	14
3,6			0	22	39
4				0	29
5					0

$\xrightarrow{(2,5)}$

	1	2,5	3,6	4
1	0	34	23	37
2,5		0	39	29
3,6			0	22
4				0

$\xrightarrow{((3,6),4)}$

	1	2,5	3,6,4
1	0	34	37
2,5		0	39
3,6,4			0

Maximum Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	37	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	23	37	34
2		0	25	20	14
3,6			0	22	39
4				0	29
5					0

$\xrightarrow{(2,5)}$

	1	2,5	3,6	4
1	0	34	23	37
2,5		0	39	29
3,6			0	22
4				0

$\xrightarrow{((3,6),4)}$

	1	2,5	3,6,4
1	0	34	37
2,5		0	39
3,6,4			0

$\xrightarrow{(1,(2,5))}$

	1,2,5	3,6,4
1,2,5	0	39
3,6,4		0

Maximum Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	37	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	23	37	34
2		0	25	20	14
3,6			0	22	39
4				0	29
5					0

$\xrightarrow{(2,5)}$

	1	2,5	3,6	4
1	0	34	23	37
2,5		0	39	29
3,6			0	22
4				0

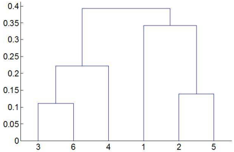
$\xrightarrow{((3,6),4)}$

	1	2,5	3,6,4
1	0	34	37
2,5		0	39
3,6,4			0

$\xrightarrow{(1,(2,5))}$

	1,2,5	3,6,4
1,2,5	0	39
3,6,4		0

$\rightarrow ((1,(2,5)),((3,6),4))$



Average Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	33	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

Average Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	33	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	23	33	34
2		0	20	20	14
3,6			0	19	35
4				0	29
5					0

Average Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	33	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	23	33	34
2		0	20	20	14
3,6			0	19	35
4				0	29
5					0

$\xrightarrow{(2,5)}$

	1	2,5	3,6	4
1	0	29	23	33
2,5		0	27	25
3,6			0	19
4				0

Average Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	33	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	23	33	34
2		0	20	20	14
3,6			0	19	35
4				0	29
5					0

$\xrightarrow{(2,5)}$

	1	2,5	3,6	4
1	0	29	23	33
2,5		0	27	25
3,6			0	19
4				0

$\xrightarrow{((3,6),4)}$

	1	2,5	3,6,4
1	0	29	26
2,5		0	26
3,6,4			0

Average Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	33	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	23	33	34
2		0	20	20	14
3,6			0	19	35
4				0	29
5					0

$\xrightarrow{(2,5)}$

	1	2,5	3,6	4
1	0	29	23	33
2,5		0	27	25
3,6			0	19
4				0

$\xrightarrow{((3,6),4)}$

	1	2,5	3,6,4
1	0	29	26
2,5		0	26
3,6,4			0

$\xrightarrow{(1,((3,6),4))}$

	1,3,6,4	2,5
1,3,6,4	0	27
2,5		0

Average Pairwise Distance

	1	2	3	4	5	6
1	0	24	22	33	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	23	33	34
2		0	20	20	14
3,6			0	19	35
4				0	29
5					0

$\xrightarrow{(2,5)}$

	1	2,5	3,6	4
1	0	29	23	33
2,5		0	27	25
3,6			0	19
4				0

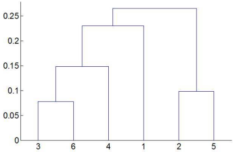
$\xrightarrow{((3,6),4)}$

	1	2,5	3,6,4
1	0	29	26
2,5		0	26
3,6,4			0

$\xrightarrow{(1,((3,6),4))}$

	1,3,6,4	2,5
1,3,6,4	0	27
2,5		0

$\rightarrow ((1,((3,6),4)),(2,5))$



Distance to Average

	1	2	3	4	5	6
1	0	24	22	33	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

Distance to Average

	1	2	3	4	5	6
1	0	24	22	33	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	22	33	34
2		0	19	20	14
3,6			0	18	33
4				0	29
5					0

Distance to Average

	1	2	3	4	5	6
1	0	24	22	33	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	22	33	34
2		0	19	20	14
3,6			0	18	33
4				0	29
5					0

$\xrightarrow{(2,5)}$

	1	2,5	3,6	4
1	0	28	22	33
2,5		0	26	23
3,6			0	18
4				0

Distance to Average

	1	2	3	4	5	6
1	0	24	22	33	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	22	33	34
2		0	19	20	14
3,6			0	18	33
4				0	29
5					0

$\xrightarrow{(2,5)}$

	1	2,5	3,6	4
1	0	28	22	33
2,5		0	26	23
3,6			0	18
4				0

$\xrightarrow{((3,6),4)}$

	1	2,5	3,6,4
1	0	28	26
2,5		0	23
3,6,4			0

Distance to Average

	1	2	3	4	5	6
1	0	24	22	33	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	22	33	34
2		0	19	20	14
3,6			0	18	33
4				0	29
5					0

$\xrightarrow{(2,5)}$

	1	2,5	3,6	4
1	0	28	22	33
2,5		0	26	23
3,6			0	18
4				0

$\xrightarrow{((3,6),4)}$

	1	2,5	3,6,4
1	0	28	26
2,5		0	23
3,6,4			0

$\xrightarrow{(((2,5),(3,6)),4)}$

	1	2,5,3,6,4
1	0	25
2,5,3,6,4		0

Distance to Average

	1	2	3	4	5	6
1	0	24	22	33	34	23
2		0	15	20	14	25
3			0	15	28	11
4				0	29	22
5					0	39
6						0

$\xrightarrow{(3,6)}$

	1	2	3,6	4	5
1	0	24	22	33	34
2		0	19	20	14
3,6			0	18	33
4				0	29
5					0

$\xrightarrow{(2,5)}$

	1	2,5	3,6	4
1	0	28	22	33
2,5		0	26	23
3,6			0	18
4				0

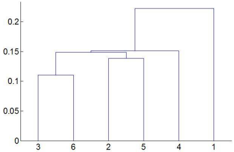
$\xrightarrow{((3,6),4)}$

	1	2,5	3,6,4
1	0	28	26
2,5		0	23
3,6,4			0

$\xrightarrow{(((2,5),(3,6)),4)}$

	1	2,5,3,6,4
1	0	25
2,5,3,6,4		0

$\rightarrow ((((2,5),(3,6)),4),1)$



Time and Space Complexities

- Assume n objects
- Initial distance matrix requires $O(n^2)$ time and space

Time and Space Complexities

- Assume n objects
- Initial distance matrix requires $O(n^2)$ time and space
- In each step, minimum distance is found out
- This requires $O(n^2)$ time
- Can be reduced to $O(n \log n)$ by using a heap

Time and Space Complexities

- Assume n objects
- Initial distance matrix requires $O(n^2)$ time and space
- In each step, minimum distance is found out
- This requires $O(n^2)$ time
- Can be reduced to $O(n \log n)$ by using a heap
- Updating the matrix requires $O(n)$ time
- Only a row or a column is updated

Time and Space Complexities

- Assume n objects
- Initial distance matrix requires $O(n^2)$ time and space
- In each step, minimum distance is found out
- This requires $O(n^2)$ time
- Can be reduced to $O(n \log n)$ by using a heap
- Updating the matrix requires $O(n)$ time
- Only a row or a column is updated
- There are $O(n)$ steps

Time and Space Complexities

- Assume n objects
- Initial distance matrix requires $O(n^2)$ time and space
- In each step, minimum distance is found out
- This requires $O(n^2)$ time
- Can be reduced to $O(n \log n)$ by using a heap
- Updating the matrix requires $O(n)$ time
- Only a row or a column is updated
- There are $O(n)$ steps
- So, total time complexity is $O(n^3)$ or $O(n^2 \log n)$

Time and Space Complexities

- Assume n objects
- Initial distance matrix requires $O(n^2)$ time and space
- In each step, minimum distance is found out
- This requires $O(n^2)$ time
- Can be reduced to $O(n \log n)$ by using a heap
- Updating the matrix requires $O(n)$ time
- Only a row or a column is updated
- There are $O(n)$ steps
- So, total time complexity is $O(n^3)$ or $O(n^2 \log n)$
- Space complexity remains $O(n^2)$

Extensions

- Combines hierarchical methods with partition-based clustering ideas to achieve better results
- Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH)
- Clustering using Representatives (CURE)
- Robust Clustering using Links (ROCK)
- Hierarchical Clustering using Dynamic Modeling (CHAMELEON)

BIRCH

- Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH)
- Each cluster has a clustering feature (CF) consisting of
  - n: *number* of points in the cluster, i.e., $\sum_{i=1}^{n} 1$
  - LS: *linear sum* of the n points, i.e., $\sum_{i=1}^{n} x_i$
  - SS: *squared sum* of the n points, i.e., $\sum_{i=1}^{n} x_i^2$
- Essentially, the 0^{th}, 1^{st} and 2^{nd} moments

BIRCH

- Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH)
- Each cluster has a clustering feature (CF) consisting of
  - n: *number* of points in the cluster, i.e., $\sum_{i=1}^{n} 1$
  - LS: *linear sum* of the n points, i.e., $\sum_{i=1}^{n} x_i$
  - SS: *squared sum* of the n points, i.e., $\sum_{i=1}^{n} x_i^2$
- Essentially, the 0^{th}, 1^{st} and 2^{nd} moments
- Example: Three 2D points in a cluster (2,5), (3,2), (4,3)
- CF $=$

BIRCH

- Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH)
- Each cluster has a clustering feature (CF) consisting of
  - n: *number* of points in the cluster, i.e., $\sum_{i=1}^{n} 1$
  - LS: *linear sum* of the n points, i.e., $\sum_{i=1}^{n} x_i$
  - SS: *squared sum* of the n points, i.e., $\sum_{i=1}^{n} x_i^2$
- Essentially, the 0^{th}, 1^{st} and 2^{nd} moments
- Example: Three 2D points in a cluster (2,5), (3,2), (4,3)
- $\text{CF} = \langle 3, (9, 10), (29, 38) \rangle$

BIRCH

- Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH)
- Each cluster has a clustering feature (CF) consisting of
  - n: *number* of points in the cluster, i.e., $\sum_{i=1}^{n} 1$
  - LS: *linear sum* of the n points, i.e., $\sum_{i=1}^{n} x_i$
  - SS: *squared sum* of the n points, i.e., $\sum_{i=1}^{n} x_i^2$
- Essentially, the 0^{th}, 1^{st} and 2^{nd} moments
- Example: Three 2D points in a cluster (2,5), (3,2), (4,3)
- $\text{CF} = \langle 3, (9, 10), (29, 38) \rangle$
- CF is *additive*, i.e., CF of a cluster formed by merging two clusters is simply the element-wise sum of their CFs

Clustering Feature

- CF is a summary of many useful statistics about the cluster
  - *Centroid* x_0

Clustering Feature

- CF is a summary of many useful statistics about the cluster
  - *Centroid* x_0

$$x_0 = \sum_{i=1}^{n} x_i / n = LS / n$$

  - *Radius* R (average distance from the centroid)

Clustering Feature

- CF is a summary of many useful statistics about the cluster
  - *Centroid* x_0

$$x_0 = \sum_{i=1}^{n} x_i / n = LS/n$$

  - *Radius* R (average distance from the centroid)

$$R = \sqrt{\sum_{i=1}^{n} (x_i - x_0)^2 / n} = \sqrt{(n.SS - LS^2)/n^2}$$

  - *Diameter* D (average distance from each other)

Clustering Feature

- CF is a summary of many useful statistics about the cluster
  - *Centroid* x_0

$$x_0 = \sum_{i=1}^{n} x_i / n = LS/n$$

  - *Radius* R (average distance from the centroid)

$$R = \sqrt{\sum_{i=1}^{n} (x_i - x_0)^2 / n} = \sqrt{(n.SS - LS^2)/n^2}$$

  - *Diameter* D (average distance from each other)

$$D = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} (x_i - x_j)^2 / n(n-1)} = \sqrt{(2n.SS - 2LS^2)/n(n-1)}$$

CF-Tree

- CF-tree is a height-balanced tree that stores CFs of clusters hierarchically
- Each leaf node represents a CF of a cluster
- Each non-leaf node stores the sum CF of its children

CF-Tree

- CF-tree is a height-balanced tree that stores CFs of clusters hierarchically
- Each leaf node represents a CF of a cluster
- Each non-leaf node stores the sum CF of its children
- A CF-tree has two parameters
  - Branching factor B: Maximum number of children a node can have
  - Threshold T: Maximum diameter of clusters stored at the leaf nodes
- These two factors control the size of a CF-tree

CF-Tree

- CF-tree is a height-balanced tree that stores CFs of clusters hierarchically
- Each leaf node represents a CF of a cluster
- Each non-leaf node stores the sum CF of its children
- A CF-tree has two parameters
  - Branching factor B: Maximum number of children a node can have
  - Threshold T: Maximum diameter of clusters stored at the leaf nodes
- These two factors control the size of a CF-tree
- If T is more, less number of leaves are required, and size is less
- If B is more, more number of clusters are combined in each level, and height is less

CF-Tree

- CF-tree is a height-balanced tree that stores CFs of clusters hierarchically
- Each leaf node represents a CF of a cluster
- Each non-leaf node stores the sum CF of its children
- A CF-tree has two parameters
  - Branching factor B: Maximum number of children a node can have
  - Threshold T: Maximum diameter of clusters stored at the leaf nodes
- These two factors control the size of a CF-tree
- If T is more, less number of leaves are required, and size is less
- If B is more, more number of clusters are combined in each level, and height is less
- CF-tree is a compact representation of the dataset

Algorithm

- BIRCH algorithm proceeds in two main phases

Algorithm

- BIRCH algorithm proceeds in two main phases
- Phase 1: CF-tree is built
- CF-tree is built incrementally, i.e., the data points are inserted in the tree one after another
- Thus, order-dependent

Algorithm

- BIRCH algorithm proceeds in two main phases
- Phase 1: CF-tree is built
- CF-tree is built incrementally, i.e., the data points are inserted in the tree one after another
- Thus, order-dependent
- Phase 2: A global clustering algorithm is applied on the leaf nodes of the CF-tree
- Uses the centroids of the leaf clusters for clustering

Algorithm

- BIRCH algorithm proceeds in two main phases
- Phase 1: CF-tree is built
- CF-tree is built incrementally, i.e., the data points are inserted in the tree one after another
- Thus, order-dependent
- Phase 2: A global clustering algorithm is applied on the leaf nodes of the CF-tree
- Uses the centroids of the leaf clusters for clustering
- Data is read once while building the tree
- Therefore, only $O(n)$ time and is fast

Algorithm

- BIRCH algorithm proceeds in two main phases
- Phase 1: CF-tree is built
- CF-tree is built incrementally, i.e., the data points are inserted in the tree one after another
- Thus, order-dependent
- Phase 2: A global clustering algorithm is applied on the leaf nodes of the CF-tree
- Uses the centroids of the leaf clusters for clustering
- Data is read once while building the tree
- Therefore, only $O(n)$ time and is fast
- Identifies spherical clusters

CURE

- Clustering Using REpresentatives (CURE)
- Hierarchical algorithm
- When merging two clusters, it avoids the extremes of looking only at the centroid or at all the points
- It tries to dampen the effect of outliers in a cluster

CURE

- Clustering Using REpresentatives (CURE)
- Hierarchical algorithm
- When merging two clusters, it avoids the extremes of looking only at the centroid or at all the points
- It tries to dampen the effect of outliers in a cluster
- For each cluster, it chooses c *well-scattered representatives*
- The representatives are then *shrunk* towards the centroid by a function of α
- Distance between two clusters is measured by minimum pairwise distance between these representative sets

Details

- Choosing c well-scattered representatives

Details

- Choosing c well-scattered representatives
  - First representative is the one at maximum distance from centroid
  - Second one is at maximum distance from first
  - And so on
- Representatives should capture the size and shape of the cluster well

Details

- Choosing c well-scattered representatives
  - First representative is the one at maximum distance from centroid
  - Second one is at maximum distance from first
  - And so on
- Representatives should capture the size and shape of the cluster well
- How to choose c?
  - If c is low, the representation is not complete
  - If c is high, noise may be captured

Details

- Choosing c well-scattered representatives
  - First representative is the one at maximum distance from centroid
  - Second one is at maximum distance from first
  - And so on
- Representatives should capture the size and shape of the cluster well
- How to choose c?
  - If c is low, the representation is not complete
  - If c is high, noise may be captured
- Shrinking representatives towards centroid x_0: $p' = p + \alpha(x_0 - p)$
  - α is the damping factor, $0 \leq \alpha \leq 1$
  - For outliers, the distance is reduced more
  - If $\alpha = 0$, no damping
  - If $\alpha = 1$, points shrink to the centroid

Details

- Choosing c well-scattered representatives
  - First representative is the one at maximum distance from centroid
  - Second one is at maximum distance from first
  - And so on
- Representatives should capture the size and shape of the cluster well
- How to choose c?
  - If c is low, the representation is not complete
  - If c is high, noise may be captured
- Shrinking representatives towards centroid x_0: $p' = p + \alpha(x_0 - p)$
  - α is the damping factor, $0 \leq \alpha \leq 1$
  - For outliers, the distance is reduced more
  - If $\alpha = 0$, no damping
  - If $\alpha = 1$, points shrink to the centroid
- Identifies non-spherical (such as elliptical) but still only convex shapes

Modifications for Large Datasets

- Sampling

Modifications for Large Datasets

- Sampling
- Choosing c representatives from a large cluster can be time consuming
- Representatives of a bigger cluster can be approximated as an union of representatives of smaller clusters
- Hence, representative is chosen from $2c$ points only

Modifications for Large Datasets

- Sampling
- Choosing c representatives from a large cluster can be time consuming
- Representatives of a bigger cluster can be approximated as an union of representatives of smaller clusters
- Hence, representative is chosen from $2c$ points only
- Partitions n points into initial p partitions, each having n/p points
- Partially cluster these partitions into q clusters each to produce $n/(p.q)$ clusters

Modifications for Large Datasets

- Sampling
- Choosing c representatives from a large cluster can be time consuming
- Representatives of a bigger cluster can be approximated as an union of representatives of smaller clusters
- Hence, representative is chosen from $2c$ points only
- Partitions n points into initial p partitions, each having n/p points
- Partially cluster these partitions into q clusters each to produce $n/(p.q)$ clusters
- These initial clusters are then clustered

ROCK

- ROBust Clustering using linKs (ROCK)
- Does *not* need vector distances
- Instead, can use *links* between clusters
- Quite applicable for categorical data

ROCK

- ROBust Clustering using linKs (ROCK)
- Does *not* need vector distances
- Instead, can use *links* between clusters
- Quite applicable for categorical data
- Two points are considered *neighbors* if their similarity exceeds a certain threshold
  - Similarity can be defined in many ways using domain knowledge
- Number of links between a pair of points is the number of *common* neighbors
- Points inside a cluster should have large number of common neighbors, and therefore, large number of links
- Hence, merge points that have more number of links

ROCK Example

- Example: documents represented as set of keywords
- Neighbor if at least two keywords is shared
- {1,2,3}, {1,2,4}, {1,2,5}, {1,3,4}, {1,3,5}, {1,4,5}, {2,3,4}, {2,3,5}, {2,4,5}, {3,4,5}
  - Number of common neighbors of {1,2,3} and {1,2,4} is

ROCK Example

- Example: documents represented as set of keywords
- Neighbor if at least two keywords is shared
- $\{1,2,3\}$, $\{1,2,4\}$, $\{1,2,5\}$, $\{1,3,4\}$, $\{1,3,5\}$, $\{1,4,5\}$, $\{2,3,4\}$, $\{2,3,5\}$, $\{2,4,5\}$, $\{3,4,5\}$
  - Number of common neighbors of $\{1,2,3\}$ and $\{1,2,4\}$ is 3
  - $\{1,2,5\}$, $\{1,3,4\}$ and $\{2,3,4\}$

ROCK Example

- Example: documents represented as set of keywords
- Neighbor if at least two keywords is shared
- {1,2,3}, {1,2,4}, {1,2,5}, {1,3,4}, {1,3,5}, {1,4,5}, {2,3,4}, {2,3,5}, {2,4,5}, {3,4,5}
  - Number of common neighbors of {1,2,3} and {1,2,4} is 3
  - {1,2,5}, {1,3,4} and {2,3,4}
- {1, 2, 3, 5}, {2, 3, 4, 5}, {1, 4}, {6}
  - Vector representations (111010, 011110, 100100, 000001)
  - Euclidean distance over vectors

ROCK Example

- Example: documents represented as set of keywords
- Neighbor if at least two keywords is shared
- $\{1,2,3\}$, $\{1,2,4\}$, $\{1,2,5\}$, $\{1,3,4\}$, $\{1,3,5\}$, $\{1,4,5\}$, $\{2,3,4\}$, $\{2,3,5\}$, $\{2,4,5\}$, $\{3,4,5\}$
  - Number of common neighbors of $\{1,2,3\}$ and $\{1,2,4\}$ is 3
  - $\{1,2,5\}$, $\{1,3,4\}$ and $\{2,3,4\}$
- $\{1, 2, 3, 5\}$, $\{2, 3, 4, 5\}$, $\{1, 4\}$, $\{6\}$
  - Vector representations (111010, 011110, 100100, 000001)
  - Euclidean distance over vectors
  - $\{1, 4\}$ and $\{6\}$ are quite close, although they have nothing in common

CHAMELEON

- Hierarchical Clustering using Dynamic Modeling (CHAMELEON)
- Clustering is based on both
  - *Interconnectivity*: How well-connected points within a cluster are
  - *Closeness* or *proximity*: How close two clusters are
- Thus, tries to incorporate the good features of both CURE and ROCK

CHAMELEON

- Hierarchical Clustering using Dynamic Modeling (CHAMELEON)
- Clustering is based on both
  - *Interconnectivity*: How well-connected points within a cluster are
  - *Closeness* or *proximity*: How close two clusters are
- Thus, tries to incorporate the good features of both CURE and ROCK
- Good in finding arbitrary shaped clusters

CHAMELEON

- Hierarchical Clustering using Dynamic Modeling (CHAMELEON)
- Clustering is based on both
  - *Interconnectivity*: How well-connected points within a cluster are
  - *Closeness* or *proximity*: How close two clusters are
- Thus, tries to incorporate the good features of both CURE and ROCK
- Good in finding arbitrary shaped clusters
- Many parameters
- Can be slow

Algorithm

- First builds a k-nearest-neighbor graph
- Sparse graph where a node (representing a point) is connected to its k most similar (i.e., nearest neighbor) points
- Edge is weighted by some (inverse) function of distance

Algorithm

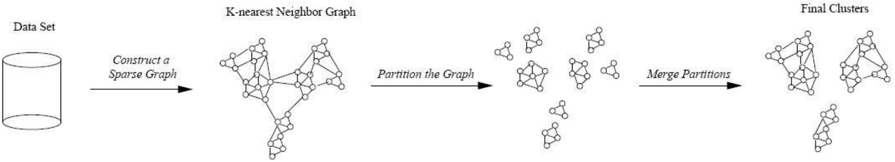
- First builds a k-nearest-neighbor graph
- Sparse graph where a node (representing a point) is connected to its k most similar (i.e., nearest neighbor) points
- Edge is weighted by some (inverse) function of distance
- Then partitions the graph into many small subclusters
- Minimizes the *edge cut*
- Therefore, two clusters are apart if the weights of their interconnections is low

Algorithm

- First builds a k-nearest-neighbor graph
- Sparse graph where a node (representing a point) is connected to its k most similar (i.e., nearest neighbor) points
- Edge is weighted by some (inverse) function of distance
- Then partitions the graph into many small subclusters
- Minimizes the *edge cut*
- Therefore, two clusters are apart if the weights of their interconnections is low
- Finally, uses a hierarchical algorithm to merge clusters
- Utilizes both interconnectivity and closeness measures

Algorithm

- First builds a k-nearest-neighbor graph
- Sparse graph where a node (representing a point) is connected to its k most similar (i.e., nearest neighbor) points
- Edge is weighted by some (inverse) function of distance
- Then partitions the graph into many small subclusters
- Minimizes the *edge cut*
- Therefore, two clusters are apart if the weights of their interconnections is low
- Finally, uses a hierarchical algorithm to merge clusters
- Utilizes both interconnectivity and closeness measures



Measures of Similarity

- Two clusters C_i and C_j

Measures of Similarity

- Two clusters C_i and C_j
- Relative interconnectivity $RI(C_i, C_j)$ is the absolute interconnectivity with respect to internal interconnectivities

$$RI(C_i, C_j) = \frac{|EC|_{\{C_i, C_j\}}}{\frac{1}{2}(|EC|_{C_i} + |EC|_{C_j})}$$

- EC is the cost of edge cut, i.e., sum of edges in the cut

Measures of Similarity

- Two clusters C_i and C_j
- Relative interconnectivity $RI(C_i, C_j)$ is the absolute interconnectivity with respect to internal interconnectivities

$$RI(C_i, C_j) = \frac{|EC|_{\{C_i, C_j\}}}{\frac{1}{2}(|EC|_{C_i} + |EC|_{C_j})}$$

- EC is the cost of edge cut, i.e., sum of edges in the cut
- Relative closeness $RC(C_i, C_j)$ is the absolute closeness with respect to internal closenesses

$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i, C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}}$$

- $\overline{S}_{EC}$ is the average weight of edges that connect pairs of vertices

Measures of Similarity

- Two clusters C_i and C_j
- Relative interconnectivity $RI(C_i, C_j)$ is the absolute interconnectivity with respect to internal interconnectivities

$$RI(C_i, C_j) = \frac{|EC|_{\{C_i, C_j\}}}{\frac{1}{2}(|EC|_{C_i} + |EC|_{C_j})}$$

- EC is the cost of edge cut, i.e., sum of edges in the cut
- Relative closeness $RC(C_i, C_j)$ is the absolute closeness with respect to internal closenesses

$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i, C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}}$$

- $\overline{S}_{EC}$ is the average weight of edges that connect pairs of vertices
- May use a combined measure or two different thresholds