패턴인식

10.4 계층 군집화

최우진

10.4 계층 군집화

- 군집 해 $C_2 = \{c_{21}, c_{22}, \cdots, c_{2n}\}$ 의 모든 군집 c_{2i}가 다른 군집 해 $C_1 = \{c_{11}, c_{12}, \cdots, c_{1k}\}$ 에 속한 군집의 부분 집합일 때 C_2는 C_1에 포함된다고 말한다. (단, n > k)
- 예) $C_2 = \{\{\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_6\}, \{\mathbf{x}_2\}, \{\mathbf{x}_4, \mathbf{x}_5\}\}$ 는 $C_1 = \{\{\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_6\}, \{\mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_5\}\}$ 에 포함된다.
- 계층 군집화(hierarchical clustering) 알고리즘의 종류

  - 응집(agglomerative) 방식

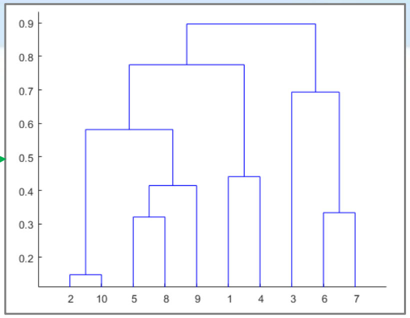
    : 작은 군집 들에서 출발하여 이들을 모아 나가는 방식

  - 분열(divisive) 방식

    : 큰 군집에서 출발하여 이들을 나누어 나가는 방식

10.4.1 응집 계층 알고리즘

알고리즘 [10.1] 응집 계층 군집화



입력: 샘플 집합 $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$

출력: 덴드로그램

알고리즘:

1. $C_0 = \{c_1 = \{\mathbf{x}_1\}, c_2 = \{\mathbf{x}_2\}, \cdots, c_N = \{\mathbf{x}_N\}\}$; // 각 샘플이 하나의 군집
2. **for** $(t = 1$ **to** $N-1)$ {
3. C_{t-1}의 모든 군집 쌍 (c_i, c_j)를 조사하여 아래를 만족하는 쌍 (c_p, c_q)를 찾아라.

   $$D(c_p, c_q) = \min_{c_i, c_j \in C_{t-1}} D(c_i, c_j) \quad \text{// 가장 가까운 쌍을 찾는 조건}$$
4. $c_r = c_p \cup c_q$; // 두 군집을 하나로 합쳐라.
5. $C_t = (C_{t-1} - c_p - c_q) \cup c_r$; // 두 군집을 제거하고 새로운 군집을 추가하라.
6. }

예제 10.3 응집 계층 알고리즘 (단일 연결 알고리즘)

- 일곱 개의 샘플이 주어진 상황

  $\mathbf{x}_1 = (18,5)^T$, $\mathbf{x}_2 = (20,9)^T$, $\mathbf{x}_3 = (20,14)^T$, $\mathbf{x}_4 = (20,17)^T$, $\mathbf{x}_5 = (5,15)^T$, $\mathbf{x}_6 = (9,15)^T$, $\mathbf{x}_7 = (6,20)^T$

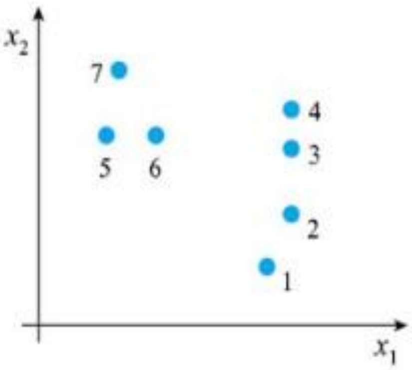


그림 10.7 군집화 예제

- (라인 1의) 초기화에 의해,

  $C_0 = \{c_1 = \{\mathbf{x}_1\}, c_2 = \{\mathbf{x}_2\}, c_3 = \{\mathbf{x}_3\}, c_4 = \{\mathbf{x}_4\}, c_5 = \{\mathbf{x}_5\}, c_6 = \{\mathbf{x}_6\}, c_7 = \{\mathbf{x}_7\}\}$

예제 10.3 응집 계층 알고리즘 (단일 연결 알고리즘)

- (라인 3)을 수행하기 위해 $D_{min}(c_i, c_j) = \min_{x_k \in c_i, y_l \in c_j} d_{kl}$ 을 사용.

- 두 점 간의 거리는 유클리디언 거리($d_{ij} = \sqrt{\sum_{k=1}^{d}(x_{ik} - x_{jk})^2}$) 을 사용.

- 루프를 반복.

$C_1 = \{c_1 = \{\mathbf{x}_1\},\ c_2 = \{\mathbf{x}_2\},\ c_3 = \{\mathbf{x}_3, \mathbf{x}_4\},\ c_4 = \{\mathbf{x}_5\},\ c_5 = \{\mathbf{x}_6\},\ c_6 = \{\mathbf{x}_7\}\}$

$C_2 = \{c_1 = \{\mathbf{x}_1\},\ c_2 = \{\mathbf{x}_2\},\ c_3 = \{\mathbf{x}_3, \mathbf{x}_4\},\ c_4 = \{\mathbf{x}_5, \mathbf{x}_6\},\ c_5 = \{\mathbf{x}_7\}\}$

$C_3 = \{c_1 = \{\mathbf{x}_1, \mathbf{x}_2\},\ c_2 = \{\mathbf{x}_3, \mathbf{x}_4\},\ c_3 = \{\mathbf{x}_5, \mathbf{x}_6\},\ c_4 = \{\mathbf{x}_7\}\}$

$C_4 = \{c_1 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\},\ c_2 = \{\mathbf{x}_5, \mathbf{x}_6\},\ c_3 = \{\mathbf{x}_7\}\}$

$C_5 = \{c_1 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\},\ c_2 = \{\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7\}\}$

$C_6 = \{c_1 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7\}\}$

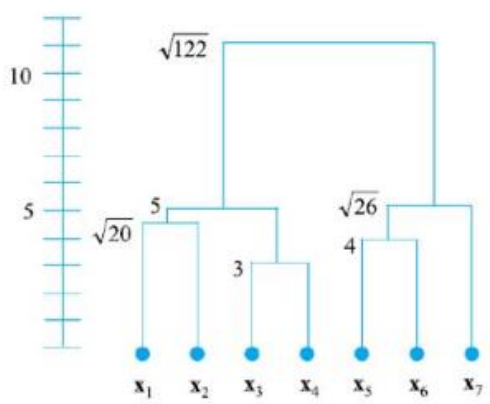


그림 10.8 단일 연결 알고리즘이 만드는 덴드로그램

예제 10.3 응집 계층 알고리즘 (단일 연결 알고리즘)

- 코드

```
def agg(X):
    c = []
    for x in X:
        c.append([x])
    num_c = len(c)
    print("초기 값: ", c)
    print("-"*100)

    for i in range(num_c - 1):
        dist_c = 99
        min_idx_i = 0
        min_idx_j = 0
        for idx_i, c_i in enumerate(c):
            for idx_j, c_j in enumerate(c):
                if idx_i >= idx_j:
                    continue
                for c_i_point in c_i:
                    for c_j_point in c_j:
                        if dist_c > dist(c_i_point, c_j_point):
                            dist_c = dist(c_i_point, c_j_point)
                            min_idx_i = idx_i
                            min_idx_j = idx_j
        print(i+1,"번째 min_dist: ", dist_c)
        print(i+1,"번째 min_idx_i: ", min_idx_i+1)
        print(i+1,"번째 min_idx_j: ", min_idx_j+1)
        c[min_idx_i] = c[min_idx_i] + c[min_idx_j]
        del c[min_idx_j]
        print(i+1,"번째 c: ",c)
        print("-"*100)
```

```
def dist(x1, x2):
    result = pow((x1[0] - x2[0]), 2) + pow((x1[1] - x2[1]), 2)
    result = math.sqrt(result)
    return result
```

예제 10.3 응집 계층 알고리즘 (단일 연결 알고리즘)

- 결과

```
agg(X)
```

```
초기 값:  [[(18, 5)], [(20, 9)], [(20, 14)], [(20, 17)], [(5, 15)], [(9, 15)], [(6, 20)]]
------------------------------------------------------------------------------------------
1 번째 min_dist:  3.0
1 번째 min_idx_i:  3
1 번째 min_idx_j:  4
1 번째 c:  [[(18, 5)], [(20, 9)], [(20, 14), (20, 17)], [(5, 15)], [(9, 15)], [(6, 20)]]
------------------------------------------------------------------------------------------
2 번째 min_dist:  4.0
2 번째 min_idx_i:  4
2 번째 min_idx_j:  5
2 번째 c:  [[(18, 5)], [(20, 9)], [(20, 14), (20, 17)], [(5, 15), (9, 15)], [(6, 20)]]
------------------------------------------------------------------------------------------
3 번째 min_dist:  4.47213595499958
3 번째 min_idx_i:  1
3 번째 min_idx_j:  2
3 번째 c:  [[(18, 5), (20, 9)], [(20, 14), (20, 17)], [(5, 15), (9, 15)], [(6, 20)]]
------------------------------------------------------------------------------------------
```

예제 10.3 응집 계층 알고리즘 (단일 연결 알고리즘)

- 결과

```
3 번째 min_dist:  4.47213595499958
3 번째 min_idx_i:  1
3 번째 min_idx_j:  2
3 번째 c:  [[(18, 5), (20, 9)], [(20, 14), (20, 17)], [(5, 15), (9, 15)], [(6, 20)]]
--------------------------------------------------------------------------------------
4 번째 min_dist:  5.0
4 번째 min_idx_i:  1
4 번째 min_idx_j:  2
4 번째 c:  [[(18, 5), (20, 9), (20, 14), (20, 17)], [(5, 15), (9, 15)], [(6, 20)]]
--------------------------------------------------------------------------------------
5 번째 min_dist:  5.0990195135927845
5 번째 min_idx_i:  2
5 번째 min_idx_j:  3
5 번째 c:  [[(18, 5), (20, 9), (20, 14), (20, 17)], [(5, 15), (9, 15), (6, 20)]]
--------------------------------------------------------------------------------------
6 번째 min_dist:  11.045361017187261
6 번째 min_idx_i:  1
6 번째 min_idx_j:  2
6 번째 c:  [[(18, 5), (20, 9), (20, 14), (20, 17), (5, 15), (9, 15), (6, 20)]]
--------------------------------------------------------------------------------------
```

예제 10.4 응집 계층 알고리즘 (완전 연결 알고리즘)

- (라인 3)을 수행하기 위해 $\boldsymbol{D_{max}(c_i, c_j)} = \max_{x_k \in c_i,\, y_l \in c_j} \boldsymbol{d_{kl}}$ 을 사용.

- 두 점 간의 거리는 유클리디언 거리($\boldsymbol{d_{ij}} = \sqrt{\sum_{k=1}^{d}(\boldsymbol{x_{ik}} - \boldsymbol{x_{jk}})^2}$) 을 사용.

- 루프를 반복.

$C_1 = \{c_1 = \{\mathbf{x}_1\},\ c_2 = \{\mathbf{x}_2\},\ c_3 = \{\mathbf{x}_3, \mathbf{x}_4\},\ c_4 = \{\mathbf{x}_5\},\ c_5 = \{\mathbf{x}_6\},\ c_6 = \{\mathbf{x}_7\}\}$

$C_2 = \{c_1 = \{\mathbf{x}_1\},\ c_2 = \{\mathbf{x}_2\},\ c_3 = \{\mathbf{x}_3, \mathbf{x}_4\},\ c_4 = \{\mathbf{x}_5, \mathbf{x}_6\},\ c_5 = \{\mathbf{x}_7\}\}$

$C_3 = \{c_1 = \{\mathbf{x}_1, \mathbf{x}_2\},\ c_2 = \{\mathbf{x}_3, \mathbf{x}_4\},\ c_3 = \{\mathbf{x}_5, \mathbf{x}_6\},\ c_4 = \{\mathbf{x}_7\}\}$

$C_4 = \{c_1 = \{x_1, x_2\}, c_2 = \{x_3, x_4\}, c_3 = \{x_5, x_6, x_7\}\}$
$C_4 = \{c_1 = \{x_1, x_2, x_3, x_4\}, c_2 = \{x_5, x_6, x_7\}\}$
$C_5 = \{c_1 = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}\}$

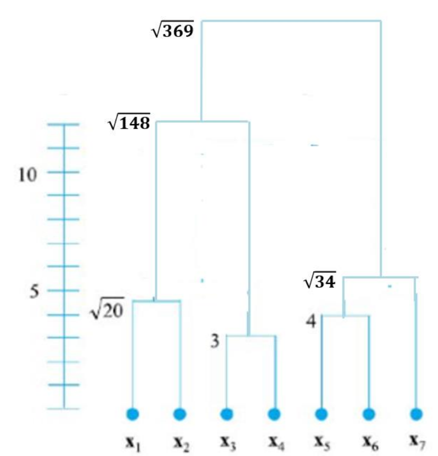


그림 10.9 완전 연결 알고리즘이 만드는 덴드로그램

예제 10.4 응집 계층 알고리즘 (완전 연결 알고리즘)

- 코드

```
def agg_max(X):
    c = []
    for x in X:
        c.append([x])
    num_c = len(c)
    print("초기 값: ", c)
    print("-"*100)

    for i in range(num_c - 1):
        dist_c = 99
        min_idx_i = 0
        min_idx_j = 0
        for idx_i, c_i in enumerate(c):
            for idx_j, c_j in enumerate(c):
                if idx_i >= idx_j:
                    continue
                if dist_c > dist_max(c_i,c_j):
                    dist_c = dist_max(c_i,c_j)
                    min_idx_i = idx_i
                    min_idx_j = idx_j
        print(i+1,"번째 min_dist: ", dist_c)
        print(i+1,"번째 min_idx_i: ", min_idx_i+1)
        print(i+1,"번째 min_idx_j: ", min_idx_j+1)
        c[min_idx_i] = c[min_idx_i] + c[min_idx_j]
        del c[min_idx_j]
        print(i+1,"번째 c: ",c)
        print("-"*100)
```

```
def dist_max(c1, c2):
    dist_c = dist(c1[0], c2[0])
    for c1_point in c1:
        for c2_point in c2:
            if dist_c < dist(c1_point, c2_point):
                dist_c = dist(c1_point, c2_point)
    return dist_c
```

예제 10.4 응집 계층 알고리즘 (완전 연결 알고리즘)

- **결과**

```
agg_max(X)
```

```
초기 값:  [[(18, 5)], [(20, 9)], [(20, 14)], [(20, 17)], [(5, 15)], [(9, 15)], [(6, 20)]]
------------------------------------------------------------------------------------------
1 번째 min_dist:  3.0
1 번째 min_idx_i:  3
1 번째 min_idx_j:  4
1 번째 c:  [[(18, 5)], [(20, 9)], [(20, 14), (20, 17)], [(5, 15)], [(9, 15)], [(6, 20)]]
------------------------------------------------------------------------------------------
2 번째 min_dist:  4.0
2 번째 min_idx_i:  4
2 번째 min_idx_j:  5
2 번째 c:  [[(18, 5)], [(20, 9)], [(20, 14), (20, 17)], [(5, 15), (9, 15)], [(6, 20)]]
------------------------------------------------------------------------------------------
3 번째 min_dist:  4.47213595499958
3 번째 min_idx_i:  1
3 번째 min_idx_j:  2
3 번째 c:  [[(18, 5), (20, 9)], [(20, 14), (20, 17)], [(5, 15), (9, 15)], [(6, 20)]]
------------------------------------------------------------------------------------------
```

예제 10.4 응집 계층 알고리즘 (완전 연결 알고리즘)

- **결과**

```
3 번째 min_dist:  4.47213595499958
3 번째 min_idx_i:  1
3 번째 min_idx_j:  2
3 번째 c:  [[(18, 5), (20, 9)], [(20, 14), (20, 17)], [(5, 15), (9, 15)], [(6, 20)]]
------------------------------------------------------------------------------------------
4 번째 min_dist:  5.830951894845301
4 번째 min_idx_i:  3
4 번째 min_idx_j:  4
4 번째 c:  [[(18, 5), (20, 9)], [(20, 14), (20, 17)], [(5, 15), (9, 15), (6, 20)]]
------------------------------------------------------------------------------------------
5 번째 min_dist:  12.165525060596439
5 번째 min_idx_i:  1
5 번째 min_idx_j:  2
5 번째 c:  [[(18, 5), (20, 9), (20, 14), (20, 17)], [(5, 15), (9, 15), (6, 20)]]
------------------------------------------------------------------------------------------
6 번째 min_dist:  19.209372712298546
6 번째 min_idx_i:  1
6 번째 min_idx_j:  2
6 번째 c:  [[(18, 5), (20, 9), (20, 14), (20, 17), (5, 15), (9, 15), (6, 20)]]
------------------------------------------------------------------------------------------
```

10.4.1 응집 계층 알고리즘

- 응집 계층 알고리즘 종류
  - 단일 연결(single linkage) 알고리즘: D_{min} 사용 (예제 10.3)
  - 완전 연결(complete linkage) 알고리즘: D_{max} 사용
  - 평균 연결(average linkage) 알고리즘: D_{ave} 사용 (예제 10.4)
- 세 알고리즘의 동작 특성
  - 단일 연결은 긴 군집을 선호, 완전 연결은 둥근 군집을 선호 (평균 연결은 중간)

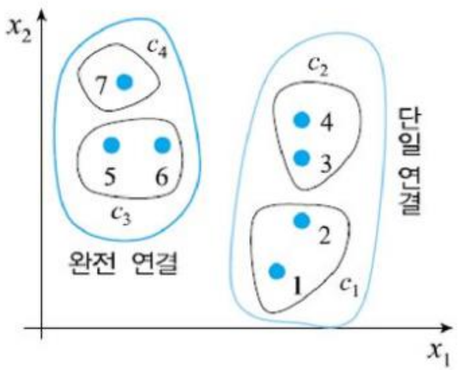


그림 10.10 단일 연결과 완전 연결 알고리즘의 동작 차이

10.4.1 응집 계층 알고리즘

- 세가지 측면에서의 부연 설명

    1. 적절한 군집의 개수를 알아내는 방법

        - 모든 군집화 알고리즘이 가지고 있는 문제.

        - 사용자 지정 또는 자동 결정.

    2. outlier 또는 noise에 대한 민감성

        - 평균 연결은 그나마 단일 연결과 완전 연결에 비해 덜 민감.

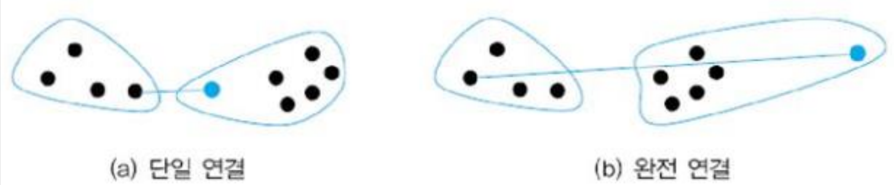


그림 10.11 외톨이에 의한 왜곡된 거리 계산

10.4.1 응집 계층 알고리즘

- 세가지 측면에서의 부연 설명

3. 시간 복잡도

$$\sum_{t=1}^{N-1} {}_{N-t+1}C_2 = \sum_{t=1}^{N-1} \frac{(N-t+1)(N-t)}{2} = \Theta(N^3)$$

- t번째 루프에서 군집 해 C_{t-1}은 N-t+1개의 군집.

- 따라서 모든 군집 쌍에 대한 거리를 계산(${}_{N-t+1}C_2$)

10.4.2 분열 계층 알고리즘

- 알고리즘

  - top-down 방식: 하나의 큰 군집을 쪼개어 가며 진행 (응집 계층 알고리즘: bottom-up 방식)

알고리즘 [10.2] **분열 계층 알고리즘**

입력: 샘플 집합 $X=\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$

출력: 덴드로그램

알고리즘:

1. $C_0=\{c_1=\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}\}$; // 모든 샘플이 하나의 군집이 되도록 초기화
2. **for** (t=1 **to** $N-1$) {
3. **for** (C_{t-1}의 모든 군집 c_i에 대하여)
4. c_i의 모든 이진 분할 중에 거리가 가장 먼 것을 찾아라.
5. 라인 3-4에서 찾은 t 개의 이진 분할을 비교하여, 가장 먼 거리를 가진 군집 c_q를 찾고 그것의 이진 분할을 c_q^1과 c_q^2라 한다.
6. $C_t=(C_{t-1}-c_q)\cup\{c_q^1, c_q^2\}$; // c_q를 제거하고 두 개의 새로운 군집을 추가
7. }

10.4.2 분열 계층 알고리즘

- 특징

  - (라인 4): 지수적 시간 복잡도

  - 군집 c_i가 n 개의 샘플을 가지면, 이진 분할의 수: $2^{n-1}-1$

  - 성능적으로 응집 방식이랑 차이가 없어서 잘 사용하지 않음

10.4 계층 군집화

감사합니다