

10. 군집화

- 패턴 인식 문제로 공식화 가능
 - 고객이 샘플이 되고 샘플은 특징 벡터 $x = (x_1, x_2, \dots, x_d)^T$ 로 표현
 - 고객의 직업이나 월평균, 구매액 등이 특징이 됨
 - 유사한(거리가 가까운) 샘플 집합을 군집이라 함
- 군집화 구현에는 두 가지 필요
 - 1) 거리 측도
 - 2) 유사한 샘플을 군집으로 만드는 알고리즘
- 지도 학습과 비지도 학습
 - 지도 학습
 - 2~7장에서 공부한 분류기 학습(MLP, SVM, HMM 등)
 - 각 샘플이 그가 속한 뿌리를 알고 있다.
(훈련 집합 $X = \{(x_1, t_1), (x_2, t_2), \dots, (x_N, t_N)\}$ 으로 표기)
 - 비지도 학습
 - 샘플은 뿌리 정보가 없다. ($X = \{x_1, x_2, \dots, x_N\}$ 으로 표기)
 - 군집화는 비지도 학습에 해당
 - 군집이 몇개인지 모르는 경우도 많다.
 - 군집화를 뿌리 발견 작업이라고도 부른다.

- 군집화의 특성
- 주관성: 군집화 결과의 품질은 응용이 치han 상황과 요구사항에 따라 다름

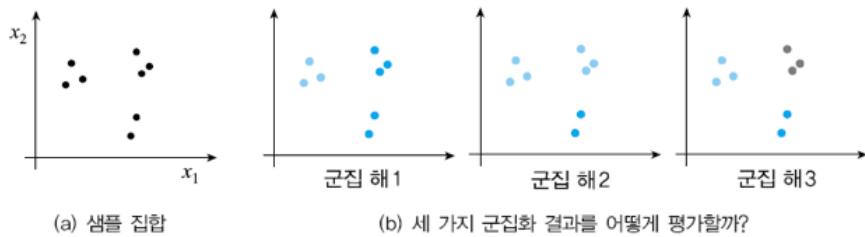


그림 10.1 군집화의 주관성

10.1 정의

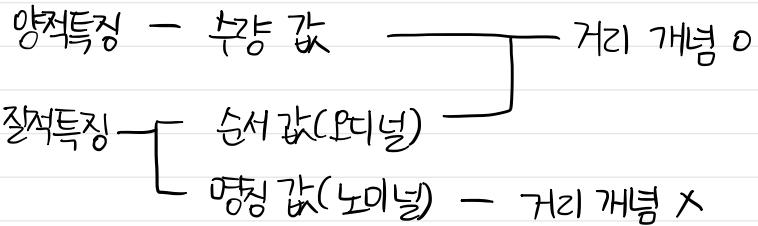
군집화란? 샘플 집합 $X = \{x_1, x_2, \dots, x_n\}$ 이 주어졌다고 하자. x_i 는 i 번째 샘플로 d 차원 특징 벡터 $x_i = (x_{i1}, x_{i2}, \dots, x_{id})^T$ 로 표기한다. 이 입력에 대해 아래 조건을 만족하는 k개의 군집으로 구성되는 군집 해(Clustering Solution) $C = \{C_1, C_2, \dots, C_K\}$ 를 찾으라. 보통 군집의 개수 k는 N에 비해 매우 작다. 상황에 따라 k가 주어지는 경우도 있고 그렇지 않고 이 값을 찾아야 하는 경우도 있다.

$$\left. \begin{array}{l} C_i \neq \emptyset, i=1, \dots, K \\ \cup_{i=1, K} C_i = X \\ C_i \cap C_j = \emptyset, i \neq j \end{array} \right\} \begin{array}{l} \text{모든 군집이 하나 이상의 샘플을 갖도록 한다.} \\ \text{모든 샘플이 하나의 군집에 포함되도록 한다.} \end{array}$$

10.2 거리와 유사도

군집화가 추구하는 본질적인 목표는 같은 군집 내의 샘플은 서로 가깝고, 다른 군집에 속한 샘플 사이의 거리는 멀게 하는 것이다. 따라서, 군집화에서 거리 개념은 매우 중요하다. 주어진 문제에 적합한 거리 측정 방법을 선택하는 것이 중요하다.

10.2.1 특징값의 종류



예제 10.1 온라인 쇼핑몰의 개인화 홍보

고객 레코드 (샘플)을 12개의 특징으로 표현한다고 하자.

$x = (\text{나이}, \text{직업}, \text{연봉}, \text{성별}, \text{월평균 구매액}, \text{반품 성향}, \text{선호하는 물품 수준}, \text{의류 선호도}, \text{전자제품 선호도}, \text{식품 선호도}, \text{팬시 선호도}, \text{DVD 선호도})^T$

나이: [1,100]의 정수

직업: [1,10]의 정수 (1 = 회사원, 2 = CEO, 3 = 교사, ...)

연봉: [1,5]의 정수 (1 = 2천만원 미만, 2 = 2천~3천만원 ..., 5 = 1억 이상)

성별: [0,1]의 정수 (0 = 여자, 1 = 남자)

월평균 구매액: 평균을 계산하여 실수로 표현

반품 성향: [1,4]의 정수 (1 =년간 반품 횟수 0, 2 = 2회 미만, ...)

선호하는 물품 수준: [1,4]의 정수 (1 = 저가, 2 = 보통, 3 = 고가, 4 = 명품)

제품 선호도: [1,5]의 정수 (1 = 구매한 적 없음, ..., 5 = 아주 선호)

수량 값을 갖는 특징: 나이, 연봉, 월평균 구매액

순서 값을 " " : 반품 성향, 선호하는 물품 수준, 제품 선호도

명칭 값을 " " : 직업, 성별

10.2.2 거리와 유사도 측정

- Minkowski 거리
 - 두 점 $x_i = (x_{i1}, \dots, x_{id})^T$ 와 $x_j = (x_{j1}, \dots, x_{jd})^T$ 간의 척도
 - $P=2$ 이면 유clidean 거리, $P=1$ 이면 도시블록 거리

$$d_{ij} = \left(\sum_{k=1}^d |x_{ik} - x_{jk}|^P \right)^{1/P}$$

유clidean 거리 ($P=2$) $d_{ij} = \sqrt{\sum_{k=1}^d (x_{ik} - x_{jk})^2}$

맨하탄 거리 ($P=1$) $d_{ij} = \sum_{k=1}^d |x_{ik} - x_{jk}|$

* 어떤 상황에서는 두 점간의 거리 계산에서 그들이 속한
분포를 고려하는 것이 합리적일 수 있다. 이때는 분산이 큰
특징에게 작은 가중치를 부여해 주는 것이 적절하다 (마할라노비스 거리)

- Hamming 거리
 - 이진 특징 벡터에 적용 가능 (서로 다른 비트의 개수)
 - ex) $(1, 0, 1, 0, 0, 0, 1, 1)^T$ 과 $(1, 0, 0, 1, 0, 0, 1, 0)^T$ 의
Hamming 거리는 3

- 코사인 유사도
- 문서 검색 응용에서 주로 사용 (단어가 특징, 출현빈도가 특징 값)
- $S_{ij} = \cos \theta = \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}$
- 이진 특징 벡터의 유사도

$$S_{ij} = \frac{n_{00} + n_{11}}{n_{00} + n_{11} + n_{01} + n_{10}}$$

$$S_{ij} = \frac{n_{11}}{n_{11} + n_{01} + n_{10}}$$

- 유사도와 거리는 쉽게 상호 변환할 수 있다.

$$S_{ij} = d_{max} - d_{ij}$$

$$d_{ij} = S_{max} - S_{ij}$$

10.2.3 점 집합을 위한 거리

- 여기에서는 점과 점 집합 또는 두 점 집합간의 거리
 - 점 x_i 와 군집(점 집합) C_j 간의 거리를 $D(x_i, C_j)$ 로 표기
 - 두 군집 C_i 와 C_j 간의 거리는 $D(C_i, C_j)$ 로 표기
- 점과 군집 사이의 거리 (모든 샘플이 참여)
 - $D_{\max}(x_i, C_j) = \max_{y_k \in C_j} d_{ik}$ (10.12)
 - $D_{\min}(x_i, C_j) = \min_{y_k \in C_j} d_{ik}$ (10.13)
 - $D_{\text{ave}}(x_i, C_j) = \frac{1}{|C_j|} \sum_{y_k \in C_j} d_{ik}$ (10.14) - 평균을 취하므로
외톨이에 덜 민감

- 점과 군집 사이의 거리(대표 샘플만 참여)
 - 평균을 대표로 삼음

$$D_{\text{mean}}(x_i, C_j) = d_{i, \text{mean}}$$

여기서 $y_{\text{mean}} = \frac{1}{|C_j|} \sum_{y_k \in C_j} y_k$

- 다른 것들과 가장 가까운 샘플을 대표로 삼음

$$D_{\text{rep}}(x_i, C_j) = d_{i, \text{rep}}$$

여기서 $\leq_{y_k \in C_j} d_{\text{rep}, k} \leq \leq_{y_k \in C_j} d_{ik}, \forall y_k \in C_j$

예제 10.2 점과 군집 사이의 거리

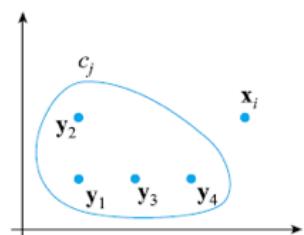
$$c_j = \{y_1 = (1, 1)^T, y_2 = (1, 2)^T, y_3 = (2, 1)^T, y_4 = (3, 1)^T\}, x_i = (4, 2)^T$$

$$D_{\max} = \max(3.162, 3.0, 2.236, 1.414) \\ = 3.162$$

$$D_{\min} = \min(3.162, 3.0, 2.236, 1.414) \\ = 1.414$$

$$D_{ave} = (3.162 + 3.0 + 2.236 + 1.414) / 4 = 2.453$$

그림 10.4 군집과 점 간의 거리



* c_j 의 대표를 사용하는 방법

$$y_{mean} = ((1, 1)^T + (1, 2)^T + (2, 1)^T + (3, 1)^T) / 4 = (1.75, 1.25)^T$$

$$D_{mean} = d_{i, mean} = 2.372$$

$$D_{rep}^1? \quad \sum_{y_k \in c_j} d_{1k} = 1.0 + 1.0 + 2.0 = 4.0$$

$$\sum_{y_k \in c_j} d_{2k} = 1.0 + 1.414 + 2.236 = 4.65$$

$$\sum_{y_k \in c_j} d_{3k} = 1.0 + 1.414 + 1.0 = \underline{\underline{3.414}}$$

$$\sum_{y_k \in c_j} d_{4k} = 2.0 + 2.236 + 2.0 = 6.236$$

$$D_{rep}(x_i, c_j) = d_{i, rep} = 2.236$$

• 두 군집 사이의 거리

- d_{K_1} 는 x_K 와 y_i , y_j 간의 거리 (x_K 는 C_i , y_i , y_j 는 C_j 에 속한 샘플)

$$D_{\max}(C_i, C_j) = \max_{x_k \in C_j, y_l \in C_j} d_{kl}$$

$$D_{\min}(C_i, C_j) = \min_{x_k \in C_j, y_l \in C_j} d_{kl}$$

$$D_{ave}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x_k \in C_i} \sum_{y_l \in C_j} d_{kl}$$

$$D_{\text{mean}}(C_i, C_j) = d_{\text{mean}_1, \text{mean}_2}$$

$$\text{여기서 } x_{\text{mean}_1} = \frac{1}{|C_i|} \sum_{x_k \in C_i} x_k, y_{\text{mean}_2} = \frac{1}{|C_j|} \sum_{y_l \in C_j} y_l$$

$$D_{\text{rep}}(C_i, C_j) = d_{\text{rep}_1, \text{rep}_2}$$

$$\text{여기서 } \sum_{x_k \in C_i} d_{\text{rep}_1, k} \leq \sum_{x_k \in C_i} d_{pk},$$

$$\forall x_p \in C_i, \sum_{y_l \in C_j} d_{\text{rep}_2, l} \leq \sum_{y_l \in C_j} d_{pl}, \forall y_p \in C_j$$

10.2.4 동적 거리

- 샘플마다 특징 벡터의 크기가 다른 경우
 - ex) 온라인 필기 인식, DNA 열
 - (6.3 절의) 교정 거리 활용

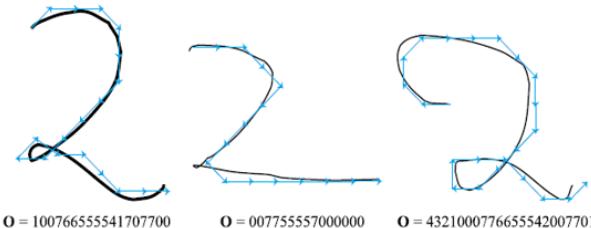


그림 7.5 온라인 필기 숫자 인식에서 부류 2의 출현 샘플들

10.3 군집화 알고리즘의 분류

- 매우 다양한 알고리즘
 - 군집화 문제의 본질적인 성질에 기인 (주관이 많이 개입되는 성질)

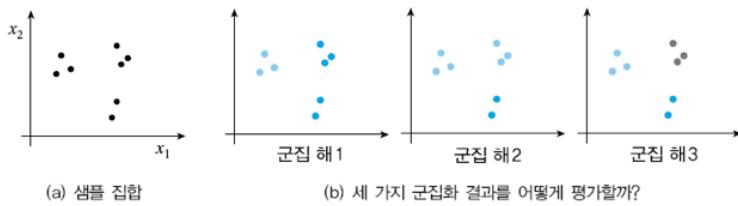


그림 10.1 군집화의 주관성

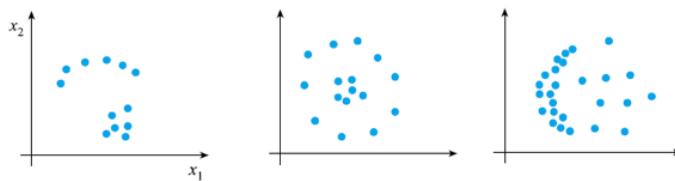


그림 10.5 다양한 군집 상황

• 분류체계

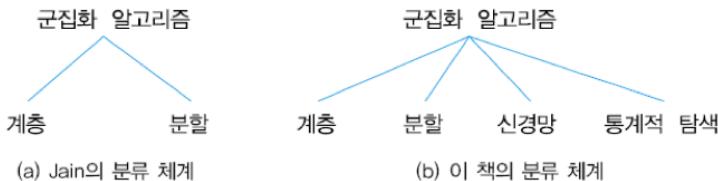


그림 10.6 군집화 알고리즘의 분류

• 분류체계

- 계층 군집화: 군집 결과를 계층을 나타내는 덴드로그램으로 표현
- 분할 군집화: 각 샘플을 군집에 배정하는 연산 사용
- 신경망: SOM, ART
- 통계적 탐색: 사뮬레이티드 어닐링, 유전 알고리즘 등

10. 4 계층 군집화



계층 군집화 알고리즘 (hierarchical clustering)은 이러한 군집 하위 포함 관계를 군집화 결과로 출력한다. 이들을 모아 나가는 응집 (agglomerative) 방식과 큰 군집에서 출발하여 이들을 나누어 나가는 분열 (divisive) 방식이 있다.

- 샘플 각각이 군집이 되어 출발, 유사한 군집을 모으는 작업을 반복
 - 출력은 군집화 결과를 트리로 표현한 덴드로그램

10.4.1 응집 계층 알고리즘

알고리즘 [10.1]

응집 계층 군집화

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

출력: 덴드로그램

알고리즘:

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

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

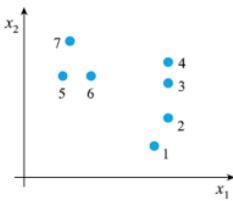


그림 10.7 군집화 예제

(간선 1의) 초기화에 의해,

$$C_0 = \{C_1 = \{\mathbf{x}_1\}, C_2 = \{\mathbf{x}_2\}, C_3 = \{\mathbf{x}_3\}, \dots, C_7 = \{\mathbf{x}_7\}\}$$

C_0 에 속한 일곱 개의 군집이 이루는 쌍은 모두 $|C_2| \geq 1$
이 중 가장 작은 거리 쌍은 $D_{\min} = 3$ 인 (C_3, C_4)이다.

$$C_1 = \{C_1 = \{x_1\}, C_2 = \{x_2\}, C_3 = \{x_3, x_4\}, C_4 = \{x_5\}, \\ C_5 = \{x_6\}, C_6 = \{x_7\}\}$$

균집의 쌍은 총 6 $C_2 = 15$

거리가 가장 작은 쌍은 $D_{min} = 40$ (C_4, C_5)

$$C_2 = \{C_1 = \{x_1\}, C_2 = \{x_2\}, C_3 = \{x_3, x_4\}, C_4 = \{x_5, x_6\}, C_5 = \{x_7\}\}$$

$$C_3 = \{C_1 = \{x_1, x_2\}, C_2 = \{x_3, x_4\}, C_3 = \{x_5, x_6\}, C_4 = \{x_7\}\}$$

$$C_4 = \{C_1 = \{x_1, x_2, x_3, x_4\}, C_2 = \{x_5, x_6\}, C_3 = \{x_7\}\}$$

$$C_5 = \{C_1 = \{x_1, x_2, x_3, x_4\}, C_2 = \{x_5, x_6, x_7\}\}$$

$$C_6 = \{C_1 = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}\}$$

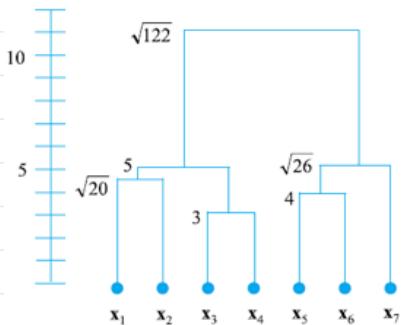


그림 10.8 단일 연결 알고리즘이 만드는 덴드로그램

- 사용하는 거리 척도에 따라 다른 이름의 알고리즘
 - D_{min} 사용하면 단일연결 (single-linkage),
 - D_{max} 사용하면 완전연결 (complete-linkage),
 - D_{ave} 사용하면 평균연결 (average-linkage)

- 세 알고리즘의 동작 특성
 - 단일 연결은 긴 군집, 완전 연결은 둘근 군집, 평균 연결은 중간을 선호

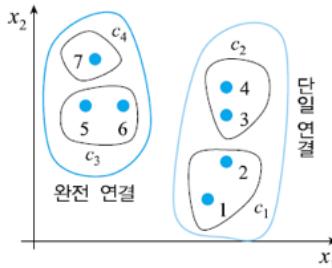


그림 10.10 단일 연결과 완전 연결 알고리즘의 동작 차이

세 가지 부연 설명

① 군집의 개수를 어떻게 알아낼까?

모든 군집화 알고리즘이 안고 있는 문제임, 사용자가 지정 또는 자동 결정

② 단일 연결과 완전 연결은 외톨이에 민감

평균 연결은 외톨이에 덜 민감



(a) 단일 연결



(b) 완전 연결

그림 10.11 외톨이에 의한 왜곡된 거리 계산

③ 계산 복잡도

t 번째 주포에서 군집 해 C_{t-1} 은 $N-t+1$ 개의 군집을 갖는다.

따라서 모든 군집 쌍에 대해 거리를 계산해야 하므로 $N-t+1$ C_2 번의 거리 계산이 필요하다. 이 연산을 $t=1$ 부터 $N-1$ 까지 반복 수행하면 총 (10.22) 만큼의 거리 계산이 필요하다.

$$\sum_{t=1}^{N-1} \sum_{t'=t+1}^{N-1} C_2 = \sum_{t=1}^{N-1} \frac{(N-t+1)(N-t)}{2} = \Theta(N^3)$$

N^3 에 정도 커지면 계산 시간이 과다, 주포 t에서 계산해 놓은 거리를 주포 $t+1$ 에서 활용할 여지가 있으므로 힘이라는 자료구조를 사용하여 효율적으로 구현 가능. $\rightarrow O(N^2 \log N)$

10.4.2 분열 계층 알고리즘

• 분열 계층은 하향 방식

알고리즘 [10.2]

분열 계층 알고리즘

입력: 샘플 집합 $X = \{x_1, x_2, \dots, x_N\}$

출력: 텐드로그램

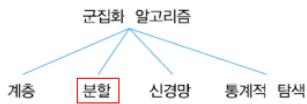
알고리즘:

1. $C_0 = \{c_1 = \{x_1, x_2, \dots, x_N\}\}; // 모든 샘플이 하나의 군집이 되도록 초기화$
2. **for** ($t=1$ to $N-1$) {
3. **for** (C_{t-1} 의 모든 군집 c_t 에 대하여)
4. c_t 의 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_q 를 찾고 그것의 이진 분할을 c_q^1 과 c_q^2 라 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. }

$$S(n, 2) = 2^{n-1}$$

지수적
복잡도

분열 방식이 응집 방식보다 특별히 두드러지는 성능 우월성이 없기 때문에 잘 사용하지 않는다.



10.5 분할 군집화

- partitional clustering
 - 순차 / k-means / MST / GMM / ...
 - 계층을 만들지 않고 하나의 군집화를 만들어 줌
 - 미리 군집의 개수 K를 알려주어야 함
 - 계층 군집화보다 빨라 대용량 데이터에 널리 사용 중

10.5.1 순차 알고리즘

• 특성

- 군집 개수를 찾아준다(대신 임계값을 설정해 주어야 함)
 - 순서에 민감하다. 빠르다 (선형 시간)

알고리즘 [10.3]

순차 알고리즘

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

출력: 군집 해 C , 군집의 개수 k

알고리즘:

- $m = 1$; // 현재 군집 개수를 뜻한다.
 - $C = \{c_1 = \{\mathbf{x}_1\}\}$; // 한 개의 군집을 가지고 출발한다.
 - for** ($i = 2$ to N) {
 - 아래 식을 이용하여 \mathbf{x}_i 와 가장 가까운 군집 c_q 를 찾는다.
$$D(\mathbf{x}_i, c_q) = \min_{1 \leq j \leq m} D(\mathbf{x}_i, c_j)$$
 - if** ($D(\mathbf{x}_i, c_q) > T$) {
 - $m++$; $c_m = \{\mathbf{x}_i\}$; $C = C \cup c_m$; // c_q 로부터 멀기 때문에 새로운 군집을 만든다.
 - else** $c_q = c_q \cup \mathbf{x}_i$; // 가까우니까 그것에 넣는다.
 - }
 - $k = m$

10.5.2 k-means 알고리즘

• 특성

- 가장 널리 쓰인다 (직관적 이해, 구현 간편)
- 군집 개수를 설정해 주어야 한다.

알고리즘 [10.4]

k-means 알고리즘

입력: 샘플 집합 $X = \{x_1, x_2, \dots, x_N\}$, 군집의 개수 k

출력: 군집 해 C

알고리즘:

1. k 개의 군집 중심 $Z = \{z_1, z_2, \dots, z_k\}$ 을 초기화 한다.
2. while (TRUE) {
 3. for ($i = 1$ to N) x_i 를 가장 가까운 군집 중심에 배정한다.
 4. if (이 배정이 이전 루프의 배정과 같음) break;
 5. for ($j = 1$ to k) z_j 에 배정된 샘플의 평균으로 z_j 를 대치한다.
 6. }

예제 10.5 k-means 알고리즘

7개 샘플을 $k=3$ 개의 군집으로 만드는 상황

$$x_1 = (18, 5)^T, x_2 = (20, 9)^T, x_3 = (20, 14)^T, x_4 = (20, 17)^T, x_5 = (5, 15)^T, x_6 = (9, 15)^T, \\ x_7 = (6, 20)^T$$

초기화에 의해 $\{x_1\} \stackrel{\text{○}}{\leftarrow} z_1$
(그림 10.12(a)), $\{x_2\} \stackrel{\text{○}}{\leftarrow} z_2$

라인 5에 의해 $z_1 = x_1 = (18, 5)^T$
(그림 10.12(b)), $z_2 = x_2 = (20, 9)^T$

$$\{x_3, x_4, x_5, x_6, x_7\} \stackrel{\text{○}}{\leftarrow} z_3$$

$$z_3 = (x_3 + x_4 + x_5 + x_6 + x_7)/5 = (12, 16.2)^T$$

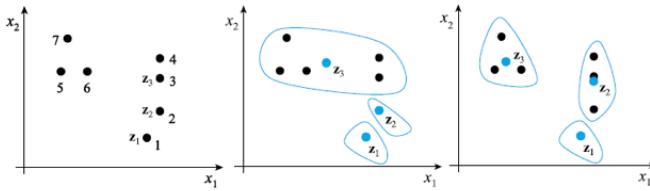


그림 10.12 k-means의 동작 예

두 번째 루프를 실행하면 $\{x_1\}$ 은 z_1

$\{x_2, x_3, x_4\}$ 은 z_2

$\{x_5, x_6, x_7\}$ 은 z_3

$$z_1 = x_1 = (18, 5)^T$$

$$z_2 = (x_2 + x_3 + x_4) / 3 = (20, 13.333)^T$$

$$z_3 = (x_5 + x_6 + x_7) / 3 = (6.667, 16.667)^T$$

세 번째 루프는 그 이전과 결과가 같으므로 멈춘다.

$$C = \{\{x_1\}, \{x_2, x_3, x_4\}, \{x_5, x_6, x_7\}\}$$

• 이론적 배경

- $(10, 23)$ 을 비용 항수로 하는 내리막 경사법의 일종

$$J(z, u) = \sum_{i=1}^N \sum_{j=1}^k u_{ji} \|x_i - z_j\|^2$$

행렬: x_i 가 군집 z_j 에 배정되었으면

$u_{ji} = 1$, 그렇지 않으면 $u_{ji} = 0$

첫 번째 루프 후의 J 를 J_1 이라 하면,

$$U = \begin{vmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{vmatrix}$$

$$\begin{aligned} J_1 &= \|x_1 - z_1\|^2 + \|x_2 - z_2\|^2 + \|x_3 - z_3\|^2 + \|x_4 - z_3\|^2 + \|x_5 - z_3\|^2 + \\ &\quad \|x_6 - z_3\|^2 + \|x_7 - z_3\|^2 = 244.8 \end{aligned}$$

두 번째 주포 J_2 ?

$$U = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$\begin{aligned} J_2 &= \|x_1 - z_1\|^2 + \|x_2 - z_2\|^2 + \|x_3 - z_2\|^2 + \|x_4 - z_2\|^2 + \\ &\quad \|x_5 - z_3\|^2 + \|x_6 - z_3\|^2 + \|x_7 - z_3\|^2 = 58.0 \end{aligned}$$

- 항상 지역 최적점으로 수렴한다. (전역 최적점 보장X)
- 초기 군집 중심에 민감 \rightarrow 다중시작 알고리즘 사용, 서로 다른 초기 군집 중심을 가지고 k-means 여러번 수행후, 가장 좋은 것을 취함
- 빠르다
- 외톨이에 민감하다 \rightarrow k-medoid를 써서 해결

샘플들 중 하나를 대표로 삼음



그림 10.13 k-means와 k-medoids의 군집 중심을 계산하는 과정의 비교
(진파랑은 k-medoids, 연파랑은 k-means)

10.5.3 모델 기반 알고리즘

- 샘플로부터 가우시안을 추정하고 그 결과에 따라 군집 배정
(가우시안 추정은 EM 알고리즘 사용)

알고리즘 [10.6]

가우시언 모델 기반 알고리즘

입력: 샘플 집합 $X = \{x_1, x_2, \dots, x_N\}$, 군집의 개수 k

출력: 군집 해 C

알고리즘:

1. X 를 가지고 EM 알고리즘을 수행하여 가우시언 $G_j, j = 1, \dots, k$ 를 추정한다.
2. (10.24)의 규칙으로 각각의 샘플을 군집에 배정한다.

x_i 를 C_q 에 배정한다.

이때 $P(Q_q | x_i) > P(G_j | x_i), j = 1, \dots, k, j \neq q$

EM 알고리즘은 속도 느리고, 지역 최소 점에 빠질 가능성을 가지고 있음.

10.6 신경망



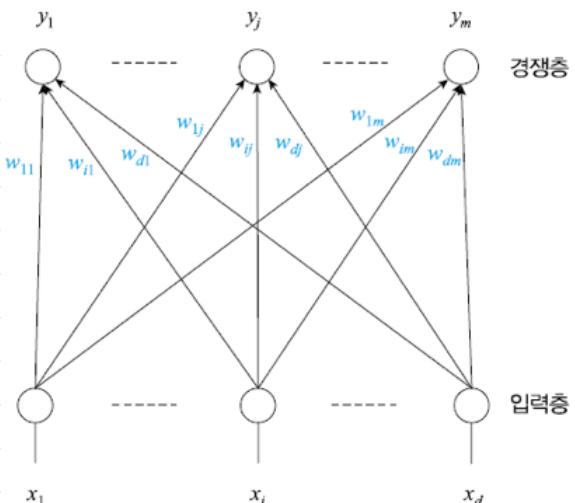
지도 학습: 퍼셉트론, 다층 퍼셉트론

비지도 학습(군집화): SOM, ART

신경망은 학습과 군집 배정 과정이 분리되어 있다. 따라서 신경망은 다른 군집화 알고리즘이 가지지 못한 독특한 장점을 가진다. 다른 알고리즘은 군집화를 마친 후 새로운 샘플이 들어오면 이것을 기존 성분에 합치고 군집화를 처음부터 다시해야 한다. 하지만 신경망은 새로운 샘플이 들어오면 학습된 신경망으로 그 샘플만 군집 배정을 해주면 된다.

10.6.1 자기 조직화 맵 (self-organizing map, SOM)

- 샘플들을 상호비교하며 스스로 군집을 조직해 냄
- 경쟁학습을 사용
 - 하나의 샘플이 입력되면 여러 대표 벡터가 경쟁
 - 가장 가까운 벡터가 승자가 되어 그것을 추함(승자 톨릭 전략)
 - 승자가 그 샘플에 적용하는 방향으로 벡터 값을 조금 변화시킴



SOM의 원리를 이해하기 위해서는 노드 y_j 로 들어오는 j 개의 가중치의 역할을 이해하는 것이 중요하다. 이들을 가중치 벡터로 보고 $w_j = (w_{1j}, w_{2j}, \dots, w_d)^T$ 와 차원이 같다. 샘플 x 가 입력되면 m 개의 가중치 벡터가 서로 경쟁하는데 그들 중 x 에 가장 가까운 벡터가 승자가 된다. 이 때

승자를 결정하기 위해서는 가중치 벡터와 샘플 벡터 간의 거리를 이용한다. 경쟁의 결과 9번재 벡터가 승자라 하면 w_9 는 x 에 적응하기 위해 자신의 값을 변화시킨다. 이 변화가 학습에 해당된다.

• SOM 학습 규칙

- (10.25)에 의해 w_{new} 는 w_{old} 보다 x 에 가깝게 된다.

$$w_{\text{new}} = w_{\text{old}} + \rho(x - w_{\text{old}})$$

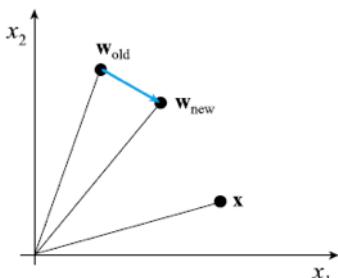


그림 10.16 SOM의 학습을 기하학적으로 해석

알고리즘 [10.7]

SOM 알고리즘

입력: 샘플 집합 $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, 최대 군집 개수 m , 학습률 ρ , 이웃 반경 r

출력: 군집 해 C

알고리즘:

1. m 개의 가중치 벡터 $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_m$ 을 초기화 한다.
2. **repeat** {
3. **for** ($n = 1$ **to** N) { // 샘플 각각에 대해
4. \mathbf{x}_n 에 가장 가까운 가중치 벡터 \mathbf{w}_q 를 찾는다. // 승자 찾기 (군집 배정)
5. **for** (q 의 이웃 노드 각각에 대해) // 학습
6. $\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} + \rho(\mathbf{x}_n - \mathbf{w}_{\text{old}})$;
7. }
8. ρ 와 r 을 조정한다. // 필요하다면
9. } **until** (stop-condition);
 // SOM 학습이 끝났으니 이제는 이것을 이용하여 군집 해를 구한다.
10. **for** ($n = 1$ **to** N)
11. \mathbf{x}_n 에 가장 가까운 가중치 벡터 \mathbf{w}_q 를 찾아 \mathbf{x}_n 을 군집 c_q 에 배정한다.
12. 군집 $c_j, j = 1, \dots, m$ 중에 비지 않을 것을 가지고 군집 해 C 를 만든다.

예제 10.6 SOM을 이용한 군집화

예제 10.3의 그림 10.7을 다시 사용해 보자. 편의상 일곱 개 샘플의 좌표를 다시 적어 주면 아래와 같다.

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

SOM의 매개 변수를 아래와 같이 설정했다 하자.

입력 노드 수 $d = 2$ (특징 벡터의 차원 수)

최대 군집 개수 $m = 3$

초기 학습률 $\rho = 0.6$, 루프를 돌 때마다 조정 $\rho_{\text{new}} = 0.8 * \rho_{\text{old}}$

이웃 반경 $r = 0$

초기 가중치 벡터는 난수 발생을 이용하여 아래와 같이 설정했다고 가정하자.

$$w_1 = (10, 12)^T, w_2 = (6, 6)^T, w_3 = (16, 14)^T$$

첫 번째 세대 시작: $x_1 = (18, 5)^T$ 로부터 w_1, w_2, w_3 까지의 유clidean 거리를 계산하고 그 중 가장 짧은 것을 찾으면 $q=3$ 이다.

$$x_1 \text{에 의해}, w_3 = w_3 + 0.6(x_1 - w_3)$$

$$= (16, 14)^T + 0.6[(18, 5)^T - (16, 14)^T]$$
$$= (17.2, 8.6)^T$$

$$x_2 \text{에 의해}, w_3 = (17.2, 8.6)^T + 0.6[(20, 9)^T - (17.2, 8.6)^T]$$
$$= (18.88, 8.64)^T$$

$$x_3 \text{에 의해}, w_3 = (18.88, 8.64)^T + 0.6[(20, 14)^T - (18.88, 8.64)^T]$$

:

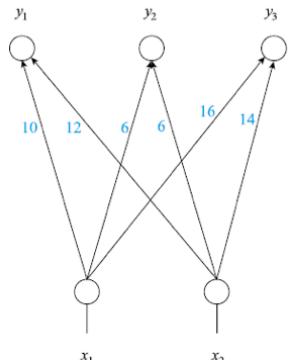
이와 같이 학습이 일어난다.

기항관계 \Rightarrow ART

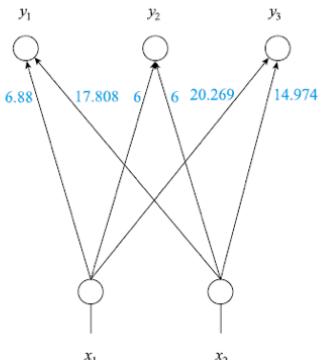
장점: **plasticity** (유연성), 새로운 샘플을 적극적으로 받아들임

단점: **stability** (안정성), 샘플이 특정 구조에 한번 배정되면

이후에도 그 구조에 계속 머물 가능성



(a) 초기 SOM



(b) 한 세대 후의 SOM

그림 10.17 예제 10.6의 SOM



10.6.2 ART

- Adaptive resonance theory (적응 공명 이론)
- SOM과 다르게 상향 가중치 b 와 하향 가중치 t 존재, t 를 이용한 검증 과정을 거침, 즉, 공명이 일어나야만 승자노드를 승자로 추구

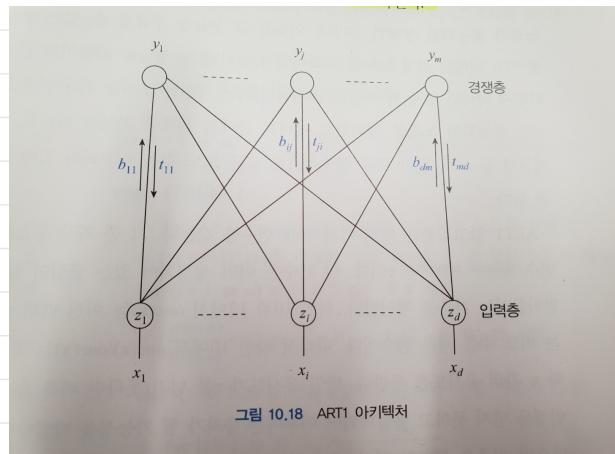


그림 10.18 ART1 아키텍처

입력: 샘플 집합 $X = \{x_1, x_2, \dots, x_N\}$ // 모든 샘플은 이진 벡터
 최대 군집 개수 m , 변수 L , 경계 변수 τ
 출력: 군집 해 C

알고리즘:

1. 상향 가중치 벡터 b_1, b_2, \dots, b_m 과 하향 가중치 벡터 t_1, t_2, \dots, t_m 을 초기화 한다.
2. **repeat** {
3. **for** ($n = 1$ to N) { // 샘플 각각에 대해
4. $\underline{z = x_n}$; // 샘플을 네트워크에 입력함
5. **for** ($j = 1$ to m) $y_j = \sum_{i=1}^d b_{ij} z_i$; // 경쟁 노드의 유사도를 계산
6. **while** (TRUE) { 내적값 가장 큰 노드 선택
7. $(y_q \geq y_j, j = 1, \dots, m)$ q 를 찾는다. // 승자 노드를 찾음
8. **if** ($y_q = -1$) $\{x_n\}$ 을 적절히 처리; **break**; } // 모든 노드가 불능임
9. **for** ($i = 1$ to d) $z_i = \underline{x_n} * t_{qi}$; // z 갱신 입력값 예측값 비교 추가 or 경계 변수 값 증정
10. **if** ($\frac{\text{one}(z)}{\text{one}(x)} < \tau$) $y_q = -1$; // 노드 q 를 불승화 시킴 유사도가 T를 넘지 못하면 비록 승자더라도 인정 받지 못하고 불능화됨, 넘으면 최종적으로 승자 인정 (입력 벡터가 이전 범위에 있는지 확인)
11. **else** { // 노드 q 학습 (상향 가중치와 하향 가중치 갱신)
12. **for** ($i = 1$ to d) $b_{iq} = \frac{L(z_i)}{L-1+\text{one}(z)}$;
13. **for** ($i = 1$ to d) $t_{qi} = z_i$;
14. **break**; // 루프 탈출
15. }
16. }
17. }
18. } **until** (stop-condition);

// ART1 학습이 끝났으니 이제는 이것을 이용하여 군집 해를 구한다.

19. **for** ($n = 1$ to N)
20. x_n 에 대한 승자 노드 y_q 를 찾아 x_n 를 군집 c_q 에 배정한다.
21. 군집 $c_j, j = 1, \dots, m$ 중에 비지 않을 것을 가지고 군집 해 C 를 만든다.

학습

분류

* $\text{one}(x)$: 이진 벡터 x 에서 1을 갖는 비트 수를 세는 함수