10장. 군집화

오일석, 패턴인식, 교보문고, 2008.

들어가는 말

재미있는 응용 시나리오를 생각해 보자. 지난 몇 년간 온라인 쇼핑몰을 성공적으로 운영했는데 이제 제 2의 도약을 꿈꾸고 있다. 그 동안 한 종류의 홍보 팜플렛을 만들어 발송했는데 이제부터는 고객의 취향을 분석하여 4~6종의 팜플렛을 만들어 맞춤 홍보를 하려 한다.[1] 일종의 개인화 personalization 홍보 전략이다. 고객에 대한 각종 정보는 데이터베이스에 저장되어 있어 이것을 기초 자료로 활용하면 된다. 고객 정보는 월평균 구매액, 선호하는 물품의 종류와 수준, 결재 방법, 반품 성향, 직업, 성별, 나이, 거주 지역 등 아주 다양하다. 하지만 수백 만 명이나 되는 고객을 어떻게 4~6개의 그룹으로 분류할 수 있을까?

- 패턴인식 문제로 공식화 가능
  - 고객이 샘플, 샘플은 특징 벡터 $\mathbf{x}=(x_1,x_2,\ldots,x_d)^T$로 표현
  - 직업, 월평균 구매액 등이 특징이 됨
  - 유사한 (거리가 가까운) 샘플 집합을 군집이라 함
- 군집화clustering 구현에는 두 가지 필요
  - 1) 거리 척도, 2)유사한 샘플을 군집으로 만드는 알고리즘

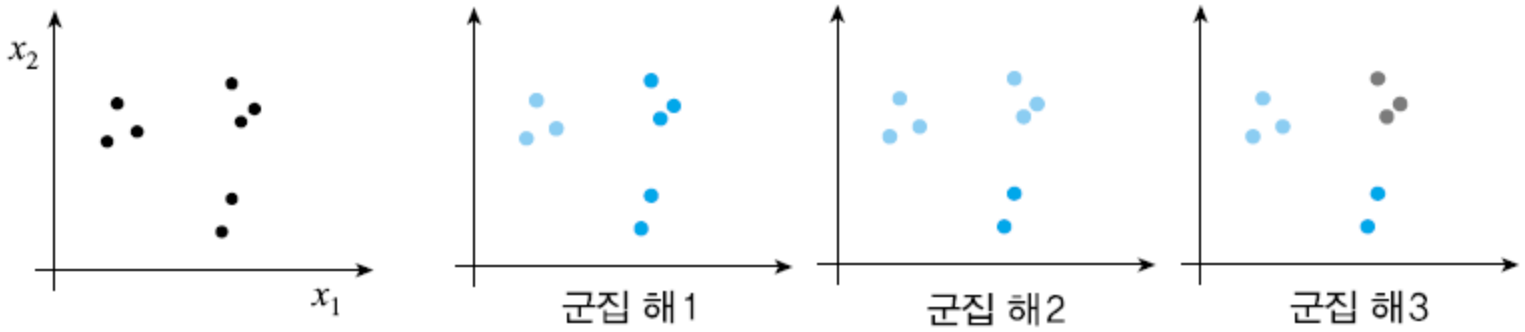
들어가는 말

- 지도 학습 과 비지도 학습
  - 지도 학습 supervised learning
    - 2~7장에서 공부한 분류기 학습 (MLP, SVM, HMM 등)
    - 각 샘플이 그가 속한 부류를 알고 있다.

      (훈련 집합 $X=\{(\mathbf{x}_1,t_1), (\mathbf{x}_2,t_2), \ldots, (\mathbf{x}_N,t_N)\}$으로 표기)
  - 비지도 학습 unsupervised learning
    - 샘플은 부류 정보가 없다. ($X=\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$으로 표기)
    - 군집화는 비지도 학습에 해당
    - 군집이 몇 개인지 모르는 경우도 많다.
    - 군집화를 부류 발견 작업이라고도 부른다.

들어가는 말

- 군집화의 특성
  - 주관성: 군집화 결과의 품질은 응용이 처한 상황과 요구 사항에 따라 다름



(a) 샘플 집합 (b) 세 가지 군집화 결과를 어떻게 평가할까?

그림 10.1 군집화의 주관성

10.1 정의

- **군집화란?** 샘플 집합 $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$이 주어졌다고 하자. $\mathbf{x}_i$는 i 번째 샘플로 d 차원 특징 벡터 $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{id})^T$로 표기한다. 이 입력에 대해 아래 조건을 만족하는 k 개의 군집으로 구성되는 군집 해 clustering solution $C = \{c_1, c_2, \cdots, c_k\}$를 찾아라. 보통 군집의 개수 k는 N에 비해 매우 작다. 상황에 따라 k가 주어지는 경우도 있고 그렇지 않고 이 값을 찾아야 하는 경우도 있다.

$$\left.\begin{array}{l} c_i \neq \varnothing, i = 1, \cdots, k \\ \cup_{i=1,k}\, c_i = X \\ c_i \cap c_j = \varnothing, i \neq j \end{array}\right\} \quad (10.1)$$

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 중요하고 여러 가지 계산 방법이 개발되어 있다. 어떤 계산 방법을 사용하느냐에 따라 군집화 결과는 달라지고 상황에 적합할 수도 있고 그렇지 않을 수도 있다. 따라서 주어진 문제에 적합한 거리 측정 방법을 선택하는 것이 매우 중요하다. 거리와 distance 유사도는 similarity 반대 개념이고 하나를 알면 다른 것은 공식을 이용하여 쉽게 계산할 수 있다.

10.2.1 특징 값의 종류

- 특징 값의 종류는 다양하다. (군집화를 공부하는데 이에 대한 이해가 필요하다.)

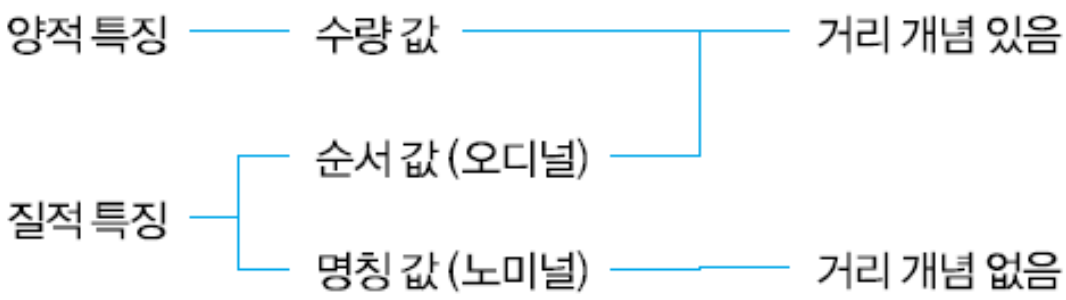


그림 10.3 특징의 유형

예제 10.1 온라인 쇼핑 몰의 개인화 홍보

고객 레코드 (샘플)을 12개의 특징으로 표현한다고 하자.

$\mathbf{x}$ = (나이, 직업, 연봉, 성별, 월평균 구매액, 반품 성향, 선호하는 물품 수준, 의류 선호도, 전자제품 선호도, 식품 선호도, 팬시 선호도, 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 갖는 특징: 반품 성향, 선호하는 물품 수준, 제품 선호도

명칭 값을 갖는 특징: 직업, 성별

10.2.2 거리와 유사도 측정

- Minkowski 거리 (10.4)
  - 두 점 $\mathbf{x}_i=(x_{i1},\ldots,x_{id})^T$와 $\mathbf{x}_j=(x_{j1},\ldots,x_{jd})^T$간의 거리 척도
  - p=2면 유클리디언 거리 (10.5), p=1이면 도시블록 거리 (10.6)

$$d_{ij}=\left(\sum_{k=1}^{d}\left|x_{ik}-x_{jk}\right|^{p}\right)^{1/p} \quad (10.4)$$

$$\text{유클리디언 거리 } (p=2) \quad d_{ij}=\sqrt{\sum_{k=1}^{d}(x_{ik}-x_{jk})^2} \quad (10.5)$$

$$\text{맨하탄 거리 } (p=1) \quad d_{ij}=\sum_{k=1}^{d}\left|x_{ik}-x_{jk}\right| \quad (10.6)$$

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

10.2.2 거리와 유사도 측정

- 코사인 유사도
  - 문서 검색 응용에서 주로 사용 (단어가 특징, 출현 빈도가 특징 값)

$$s_{ij} = \cos\theta = \frac{\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|} \tag{10.8}$$

- 이진 특징 벡터의 유사도

$$s_{ij} = \frac{n_{00} + n_{11}}{n_{00} + n_{11} + n_{01} + n_{10}} \tag{10.9}$$

$$s_{ij} = \frac{n_{11}}{n_{11} + n_{01} + n_{10}} \tag{10.10}$$

- 유사도와 거리는 쉽게 상호 변환할 수 있다.

$$s_{ij} = d_{\max} - d_{ij} \tag{10.7}$$

$$d_{ij} = s_{\max} - s_{ij} \tag{10.11}$$

10.2.3 점 집합을 위한 거리

- 여기에서는 점과 점 집합 또는 두 점 집합 간의 거리 (10.2.2 절은 두 점간의 거리 정의)
  - 점 $\mathbf{x}_i$와 군집 (점 집합) c_j 간의 거리를 $D(\mathbf{x}_i, c_j)$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)

$$D_{\max}(\mathbf{x}_i, c_j) = \max_{\mathbf{y}_k \in c_j} d_{ik} \qquad (10.12)$$

$$D_{\min}(\mathbf{x}_i, c_j) = \min_{\mathbf{y}_k \in c_j} d_{ik} \qquad (10.13)$$

$$D_{\text{ave}}(\mathbf{x}_i, c_j) = \frac{1}{|c_j|} \sum_{\mathbf{y}_k \in c_j} d_{ik} \qquad (10.14)$$

  - d_{ik}는 $\mathbf{x}_i$와 $\mathbf{y}_k$는 간의 거리 ($\mathbf{y}_k$는 c_j에 속한 샘플)
  - $D_{\max}$와 $D_{\min}$은 외톨이에outlier 민감

10.2.3 점 집합을 위한 거리

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

$$\left.\begin{aligned} & D_{\text{mean}}(\mathbf{x}_i, c_j) = d_{i,mean} \\ & \text{여기서 } \mathbf{y}_{mean} = \frac{1}{|c_j|}\sum_{\mathbf{y}_k \in c_j} \mathbf{y}_k \end{aligned}\right\} \quad (10.15)$$

  - 다른 것들과 가장 가까운 샘플을 대표로 삼음

$$\left.\begin{aligned} & D_{\text{rep}}(\mathbf{x}_i, c_j) = d_{i,rep} \\ & \text{여기서 } \sum_{\mathbf{y}_k \in c_j} d_{rep,k} \le \sum_{\mathbf{y}_k \in c_j} d_{lk}, \forall \mathbf{y}_l \in c_j \end{aligned}\right\} \quad (10.16)$$

예제 10.2 점과 군집 사이의 거리

$$c_j = \{\mathbf{y}_1 = (1,1)^T,\ \mathbf{y}_2 = (1,2)^T,\ \mathbf{y}_3 = (2,1)^T,\ \mathbf{y}_4 = (3,1)^T\},\ \mathbf{x}_i = (4,2)^T$$

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

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

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

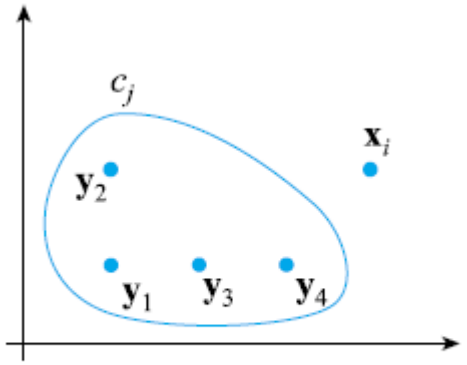


그림 10.4 군집과 점 간의 거리

$$\mathbf{y}_{\text{mean}} = ((1,1)^T + (1,2)^T + (2,1)^T + (3,1)^T)/4 = (1.75,1.25)^T$$

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

$$D_{\text{rep}}(\mathbf{x}_i, c_j) = d_{i,rep} = 2.236$$

$$\sum_{\mathbf{y}_k \in c_j} d_{1k} = 1.0+1.0+2.0 = 4.0$$

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

$$\sum_{\mathbf{y}_k \in c_j} d_{3k} = 1.0+1.414+1.0 = 3.414$$

$$\sum_{\mathbf{y}_k \in c_j} d_{4k} = 2.0+2.236+2.0 = 6.236$$

(*rep*=3이 됨)

10.2.3 점 집합을 위한 거리

- 두 군집 사이의 거리
  - d_{kl}는 $\mathbf{x}_k$와 $\mathbf{y}_l$ 간의 거리 ($\mathbf{x}_k$는 c_i, $\mathbf{y}_l$은 c_j에 속한 샘플)

$$D_{\max}(c_i, c_j) = \max_{\mathbf{x}_k \in c_i, \mathbf{y}_l \in c_j} d_{kl} \tag{10.17}$$

$$D_{\min}(c_i, c_j) = \min_{\mathbf{x}_k \in c_i, \mathbf{y}_l \in c_j} d_{kl} \tag{10.18}$$

$$D_{\text{ave}}(c_i, c_j) = \frac{1}{|c_i||c_j|} \sum_{\mathbf{x}_k \in c_i} \sum_{\mathbf{y}_l \in c_j} d_{kl} \tag{10.19}$$

$$\left.\begin{aligned} &D_{\text{mean}}(c_i, c_j) = d_{mean1,mean2} \\ &\text{여기서 } \mathbf{x}_{mean1} = \frac{1}{|c_i|} \sum_{\mathbf{x}_k \in c_i} \mathbf{x}_k, \mathbf{y}_{mean2} = \frac{1}{|c_j|} \sum_{\mathbf{y}_l \in c_j} \mathbf{y}_l \end{aligned}\right\} \tag{10.20}$$

$$\left.\begin{aligned} &D_{\text{rep}}(c_i, c_j) = d_{rep1,rep2} \\ &\text{여기서 } \sum_{\mathbf{x}_k \in c_i} d_{rep1,k} \le \sum_{\mathbf{x}_k \in c_i} d_{pk}, \forall \mathbf{x}_p \in c_i, \sum_{\mathbf{y}_l \in c_j} d_{rep2,l} \le \sum_{\mathbf{y}_l \in c_j} d_{pl}, \forall \mathbf{y}_p \in c_j \end{aligned}\right\} \tag{10.21}$$

10.2.4 동적 거리

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

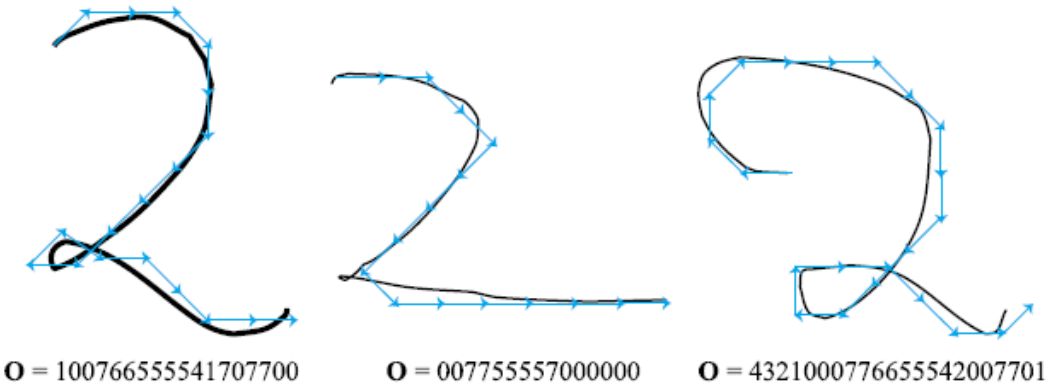


그림 7.5 온라인 필기 숫자 인식에서 부류 2의 훈련 샘플들

10.3 군집화 알고리즘의 분류

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

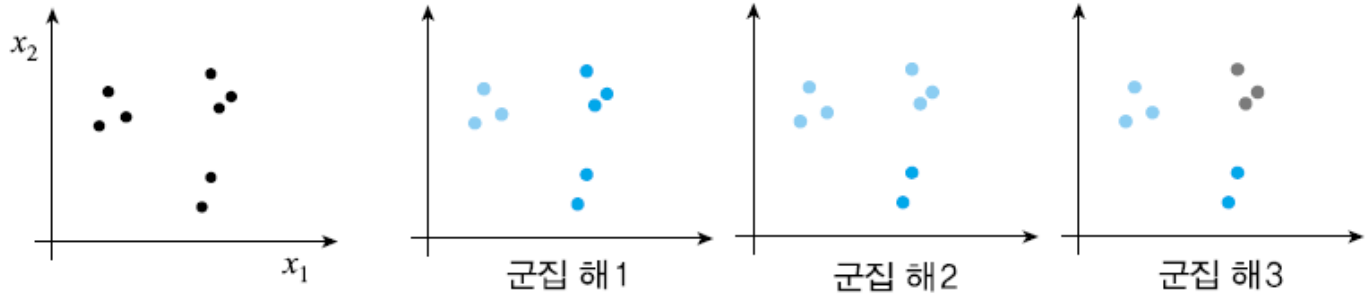


그림 10.1 군집화의 주관성

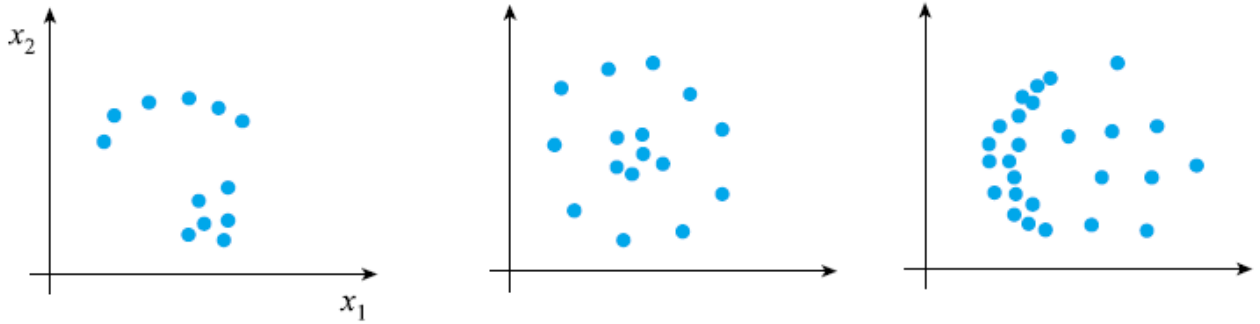
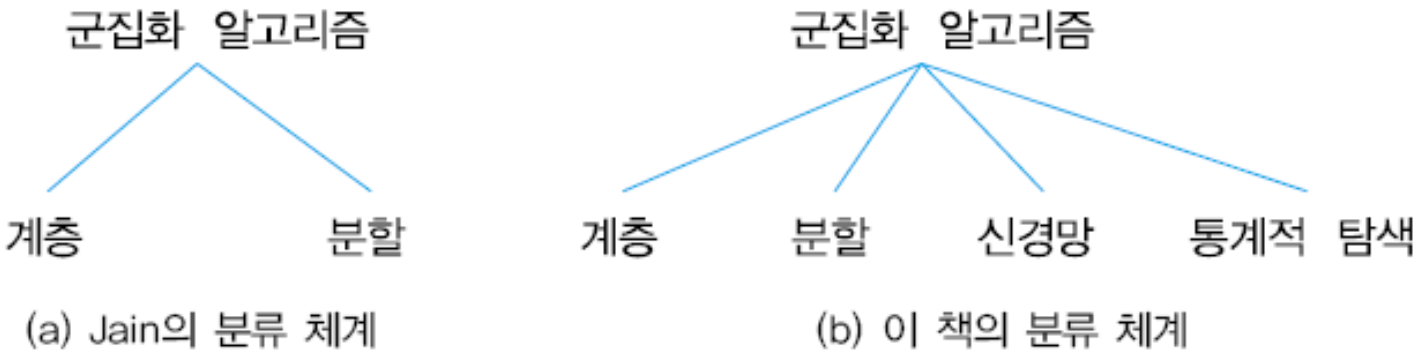


그림 10.5 다양한 군집 상황

10.3 군집화 알고리즘의 분류

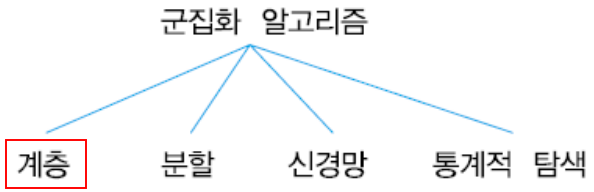
- 분류 체계
  - 계층 군집화: 군집 결과를 계층을 나타내는 덴드로그램으로 표현
  - 분할 군집화: 각 샘플을 군집에 배정하는 연산 사용
  - 신경망: SOM, ART
  - 통계적 탐색: 시뮬레이티드 어닐링, 유전 알고리즘 등



(a) Jain의 분류 체계 (b) 이 책의 분류 체계

그림 10.6 군집화 알고리즘의 분류

10.4 계층 군집화



우선 군집 해의 포함 관계를 정의하자. 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. 예를 들어 $C_2 = \{\{\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_6\}, \{\mathbf{x}_2\}, \{\mathbf{x}_4, \mathbf{x}_5\}\}$는 $C_1 = \{\{\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_6\}, \{\mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_5\}\}$에 포함된다. 물론 $n > k$이어야 한다. 계층 군집화 알고리즘은 이러한 포함 관계를 군집화 결과로 출력한다. 계층 군집화 hierarchical clustering 알고리즘에는 작은 군집들에서 출발하여 이들을 모아 나가는 **응집** agglomerative 방식과 큰 군집에서 출발하여 이들을 나누어 나가는 **분열** divisive 방식이 있다.

10.4.1 응집 계층 알고리즘

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

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.

$$D(c_p, c_q) = \min_{c_i, c_j \in C_{t-1}} D(c_i, c_j)$$ // 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

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

$\mathbf{x}_7 = (6,20)^T$

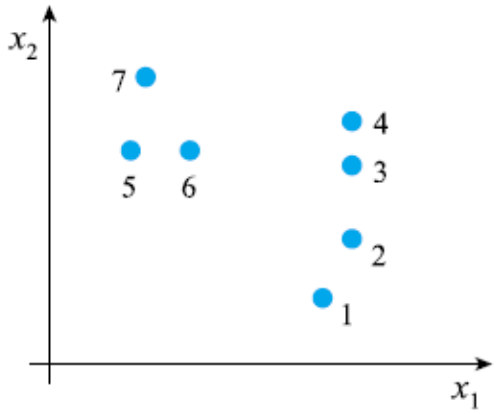


그림 10.7 군집화 예제

(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)

루프를 반복하면, (거리 척도는 유클리언 거리와 $D_{\min}$을 가정)

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

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

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

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

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

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

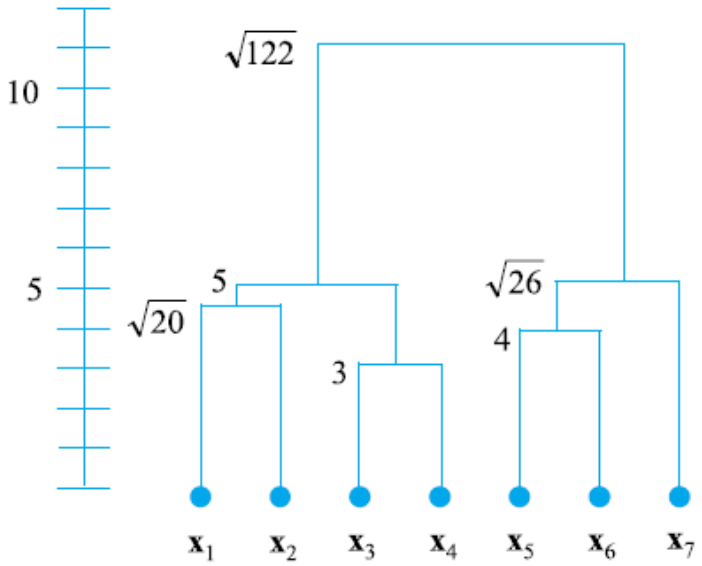


그림 10.8 단일 연결 알고리즘이 만드는 덴드로그램

연필을 들고 직접 계산해 보자.
직접 해보는 것의 힘은 생각보다 크다.

$D_{\min}$대신 $D_{\max}$를 사용하면 어떻게 될까?

10.4.1 응집 계층 알고리즘

- 사용하는 거리 척도에 따라 다른 이름의 알고리즘
  - $D_{\min}$ 사용하면 단일 연결single-linkage, $D_{\max}$ 사용하면 완전 연결complete-linkage, D_{ave} 사용하면 평균 연결average-linkage 알고리즘
- 세 알고리즘의 동작 특성
  - 단일 연결은 긴 군집, 완전 연결은 둥근 군집을 선호 (평균 연결은 중간)
  - 예) 그림 10.10

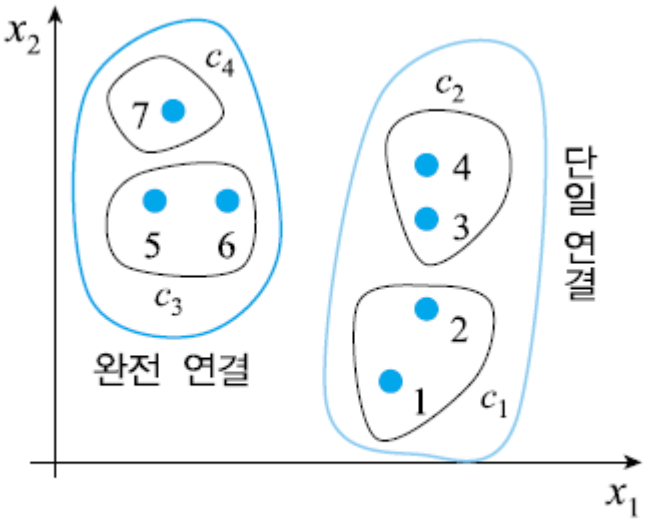


그림 10.10 단일 연결과 완전 연결 알고리즘의 동작 차이

10.4.1 응집 계층 알고리즘

- 세가지 부연 설명
  - 군집의 개수를 어떻게 알아낼까?
    - 모든 군집화 알고리즘이 안고 있는 문제임
    - 사용자가 지정 또는 자동 결정 (자동 결정은 어렵다.)
  - 단일 연결과 완전 연결은 외톨이에outlier 민감
    - 평균 연결은 외톨이에 덜 민감

(a) 단일 연결 (b) 완전 연결

그림 10.11 외톨이에 의한 왜곡된 거리 계산

  - 계산 복잡도 $$\sum_{t=1}^{N-1} {}_{N-t+1}C_2 = \sum_{t=1}^{N-1} \frac{(N-t+1)(N-t)}{2} = \Theta(N^3) \quad (10.22)$$
    - 세제곱에 비례하므로 효율적인 구현 필요
    - CURE, ROCK, CHAMELEON, BIRCH 등 [Xu05]

10.4.2 분열 계층 알고리즘

- 분열 계층은 하향 방식top-down
  - 모든 샘플이 하나의 군집으로 출발, 군집을 나누는 작업을 반복

(앞 절의 응집 계층은 상향 방식bottom-up: 샘플 각각이 군집이 되어 출발, 유사한 군집을 모으는 작업을 반복)

알고리즘 [10.2] 분열 계층 알고리즘

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

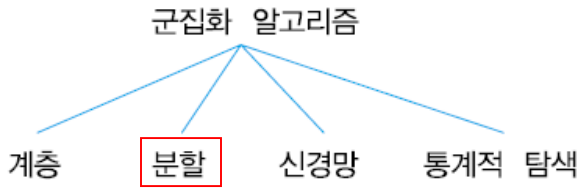
출력: 덴드로그램

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. }

지수적 복잡도

10.5 분할 군집화partitional clustering



- 다양한 알고리즘
  - 순차 알고리즘
  - k-means 알고리즘
  - MST알고리즘
  - GMM 알고리즘
  - …

10.5.1 순차 알고리즘

- 특성
  - 군집 개수를 찾아준다. (대신 임계값을 설정해주어야 한다.)
  - 순서에 민감하다. 빠르다 (선형 시간).

알고리즘 [10.3] 순차 알고리즘

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

출력: 군집 해 C, 군집의 개수 k

알고리즘:

1. $m = 1$; // 현재 군집 개수를 뜻한다.
2. $C = \{c_1 = \{\mathbf{x}_1\}\}$; // 한 개의 군집을 가지고 출발한다.
3. **for** $(i = 2$ **to** $N)$ {
4. 아래 식을 이용하여 $\mathbf{x}_i$와 가장 가까운 군집 c_q를 찾는다.

$$D(\mathbf{x}_i, c_q) = \min_{1 \le j \le m} D(\mathbf{x}_i, c_j)$$

5. **if** $(D(\mathbf{x}_i, c_q) > T)$ { m++; $c_m = \{\mathbf{x}_i\}$; $C = C \cup c_m$;}; // c_q로부터 멀기 때문에 새로운 군집을 만든다.
6. **else** $c_q = c_q \cup \mathbf{x}_i$; // 가까우니까 그것에 넣는다.
7. }
8. $k = m$;

10.5.2 *k*-means 알고리즘

- 특성
  - 가장 널리 쓰인다. (직관적 이해. 구현 간편)
  - 군집 개수를 설정해주어야 한다.

알고리즘 [10.4] *k*-means 알고리즘

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

출력: 군집 해 C

알고리즘:

1. k 개의 군집 중심 $Z = \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_k\}$를 초기화 한다.
2. **while** (TRUE) {
3. **for** ($i = 1$ **to** N) $\mathbf{x}_i$를 가장 가까운 군집 중심에 배정한다.
4. **if** (이 배정이 이전 루프의 배정과 같음) **break**;
5. **for** ($j = 1$ **to** k) $\mathbf{z}_j$에 배정된 샘플의 평균으로 $\mathbf{z}_j$를 대치한다.
6. }

예제 10.5 k-means 알고리즘

7개 샘플을 k=3개의 군집으로 만드는 상황

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

초기화에 의해 (그림 10.12(a)), $\{\mathbf{x}_1\}$은 $\mathbf{z}_1$

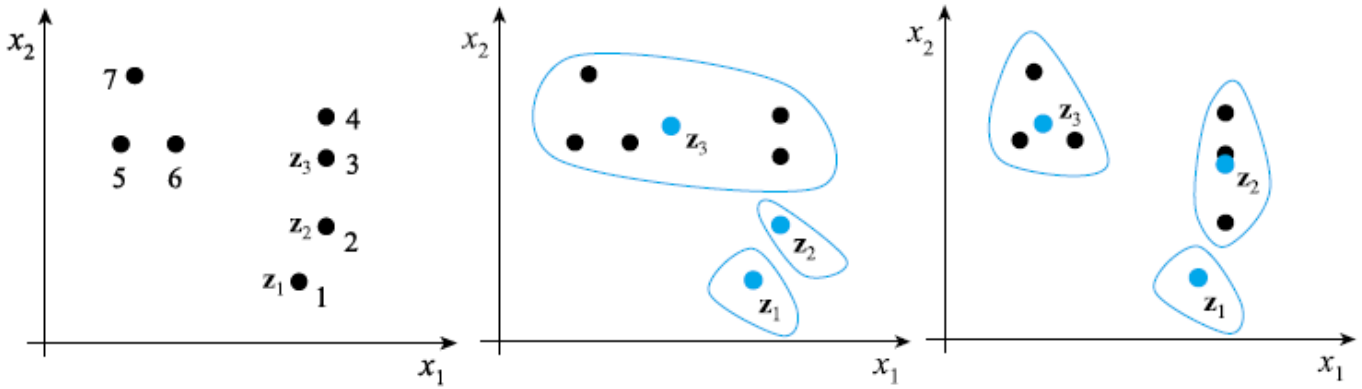
$\{\mathbf{x}_2\}$은 $\mathbf{z}_2$

$\{\mathbf{x}_3,\mathbf{x}_4,\mathbf{x}_5,\mathbf{x}_6,\mathbf{x}_7\}$은 $\mathbf{z}_3$

라인 5에 의해 (그림 10.12(b)), $\mathbf{z}_1 = \mathbf{x}_1 = (18,5)^T$

$\mathbf{z}_2 = \mathbf{x}_2 = (20,9)^T$

$\mathbf{z}_3 = (\mathbf{x}_3+\mathbf{x}_4+\mathbf{x}_5+\mathbf{x}_6+\mathbf{x}_7)/5 = (12,16.2)^T$



(a) 초기 군집 중심 (b) 첫 번째 루프 후 (c) 두 번째 루프 후

그림 10.12 k-means의 동작 예

예제 10.5 k-means 알고리즘

두 번째 루프를 실행하면 (그림 10.12(c)),

$\{\mathbf{x}_1\}$은 $\mathbf{z}_1$

$\{\mathbf{x}_2,\mathbf{x}_3,\mathbf{x}_4\}$은 $\mathbf{z}_2$

$\{\mathbf{x}_5,\mathbf{x}_6,\mathbf{x}_7\}$은 $\mathbf{z}_3$

$$\mathbf{z}_1 = \mathbf{x}_1 = (18,5)^T$$

$$\mathbf{z}_2 = (\mathbf{x}_2+\mathbf{x}_3+\mathbf{x}_4)/3 = (20,13.333)^T$$

$$\mathbf{z}_3 = (\mathbf{x}_5+\mathbf{x}_6+\mathbf{x}_7)/3 = (6.667,16.667)^T$$

세 번째 루프는 그 이전과 결과가 같다. 따라서 멈춘다.

결국 출력은

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

10.5.2 *k*-means 알고리즘

- 이론적 배경
  - (10.23)을 비용 함수로 하는 내리막 경사법의 일종

$$J(Z,\mathbf{U}) = \sum_{i=1}^{N}\sum_{j=1}^{k} u_{ji}\left\|\mathbf{x}_i - \mathbf{z}_j\right\|^2 \qquad (10.23)$$

  - 항상 지역 최적점으로 수렴한다. (전역 최적점 보장 못함)
  - 초기 군집 중심에 민감
  - 빠르다.
  - 외톨이에 민감하다. (*k*-medoids는 덜 민감)

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 가우시언을 추정하고 그 결과에 따라 군집 배정
  - 가우시언 추정은 EM 알고리즘을 사용할 수 있다.

알고리즘 [10.6] 가우시언 모델 기반 알고리즘

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

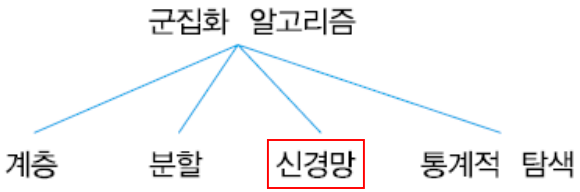
출력: 군집 해 C

알고리즘:

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

$$\left.\begin{array}{l} \mathbf{x}_i\text{를 } c_q\text{에 배정한다.} \\ \text{이때 } P(G_q \mid \mathbf{x}_i) > P(G_j \mid \mathbf{x}_i),\ j = 1, \cdots, k,\ j \neq q \end{array}\right\} \quad (10.24)$$

10.6 신경망



- 신경망
  - 지도 학습: 퍼셉트론, 다층 퍼셉트론 (4 장)
  - 비지도 학습 (군집화): SOM, ART (10 장)

10.6.1 자기 조직화 맵

- 자기 조직화 맵self-organizing map (SOM)
  - 샘플들을 상호 비교하며 스스로 군집을 조직해 냄
  - 경쟁 학습을 사용
    - 하나의 샘플이 입력되면 여러 대표 벡터가 경쟁
    - 가장 가까운 벡터가 승자가 되어 그것을 취함 (승자 독식 전략)
    - 승자는 그 샘플에 적응하는 방향으로 벡터 값을 조금 변화시킴

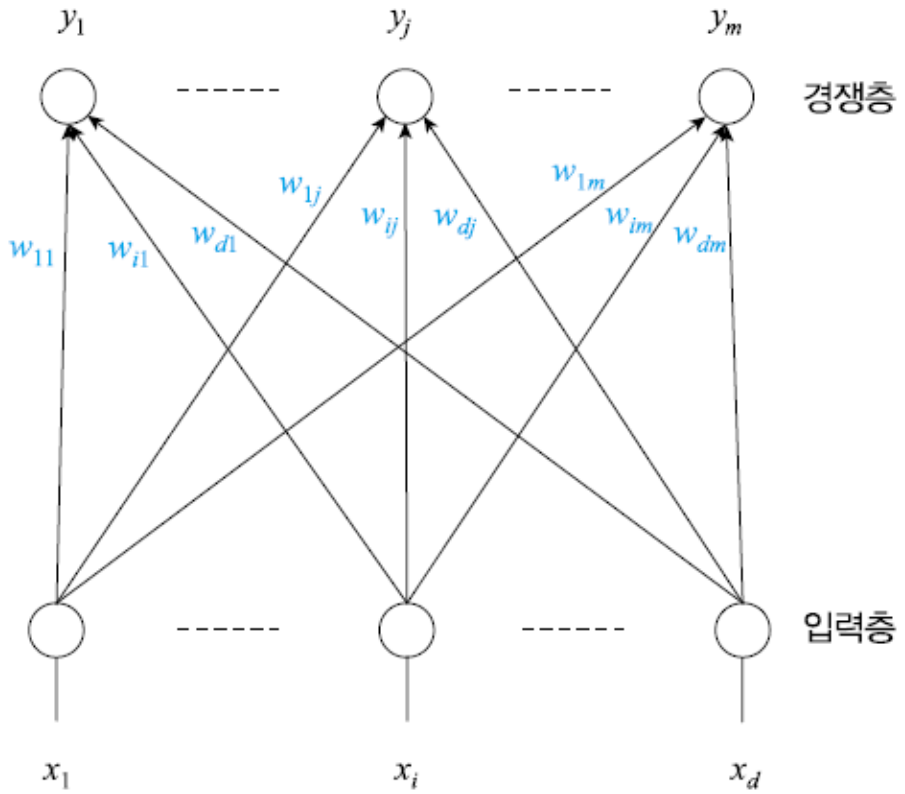


그림 10.14 SOM 아키텍처

SOM의 원리를 이해하기 위해서는 노드 y_j로 들어오는 d 개의 가중치의 역할을 이해하는 것이 매우 중요하다. 이들을 가중치 벡터로 보고 $\mathbf{w}_j = (w_{1j}, w_{2j}, \cdots, w_{dj})^{\mathrm{T}}$로 표기하자. 이 벡터는 d 차원으로서 샘플의 특징 벡터 $\mathbf{x} = (x_1, x_2, \cdots, x_d)^{\mathrm{T}}$와 차원이 같다. 샘플 $\mathbf{x}$가 입력되면 m 개의 가중치 벡터가 서로 경쟁하는데 그들 중 $\mathbf{x}$에 가장 가까운 벡터가 승자가 된다. 이때 승자를 결정하기 위해서는 가중치 벡터와 샘플 벡터 간의 거리를 이용한다. 경쟁의 결과 q 번째 벡터가 승자라 하면 $\mathbf{w}_q$는 $\mathbf{x}$에 적응하기 위해 자신의 값을 변화시킨다. 이 변화가 학습에 해당한다.

10.6.1 자기 조직화 맵

- SOM 학습 규칙
  - (10.25)에 의해 $\mathbf{w}_{new}$는 $\mathbf{w}_{old}$보다 $\mathbf{x}$에 가깝게 된다.

$$\mathbf{w}_{new} = \mathbf{w}_{old} + \rho(\mathbf{x} - \mathbf{w}_{old}) \qquad (10.25)$$

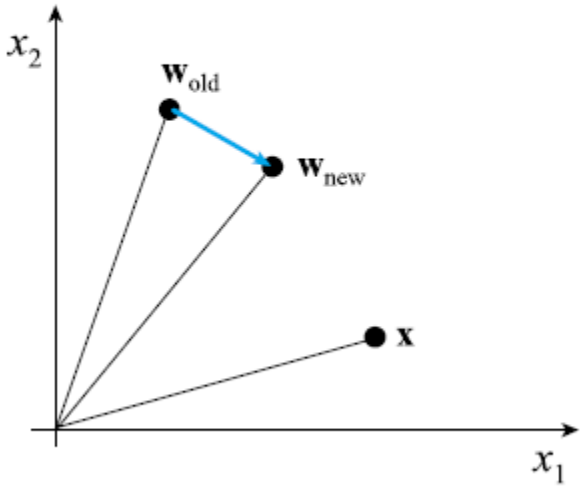


그림 10.16 SOM의 학습을 기하학적으로 해석

알고리즘 [10.7] SOM 알고리즘

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

출력: 군집 해 C

알고리즘:

1. m 개의 가중치 벡터 $\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_m$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** (q의 이웃 노드 각각에 대해) // 학습
6. $\mathbf{w}_{new} = \mathbf{w}_{old} + \rho(\mathbf{x}_n - \mathbf{w}_{old})$;
7. }
8. ρ와 r을 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_q에 배정한다.
12. 군집 c_j, $j = 1, \cdots, m$ 중에 비지 않을 것을 가지고 군집 해 C를 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.

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

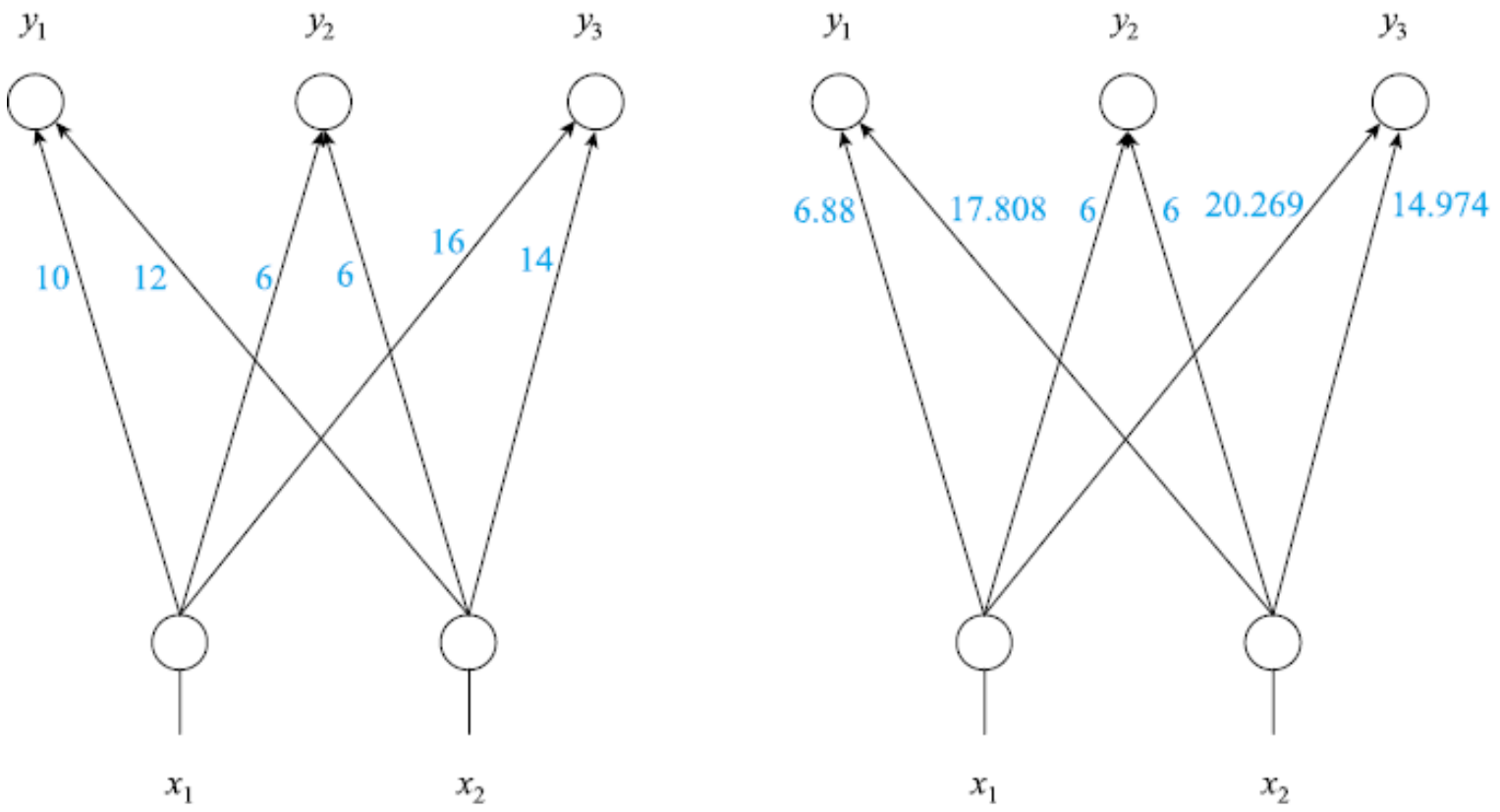
SOM의 매개 변수를 아래와 같이 설정했다 하자.

입력 노드 수 $d = 2$ (특징 벡터의 차원 수)

최대 군집 개수 $m = 3$

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

이웃 반경 $r = 0$



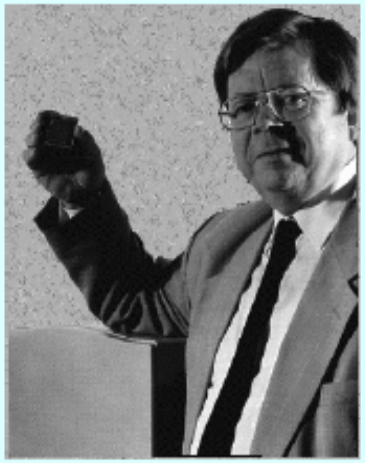
(a) 초기 SOM

(b) 한 세대 후의 SOM

그림 10.17 예제 10.6의 SOM

Teuvo Kohonen

(1934년 7월 11일 ~) 핀란드



Kohonen은 자기 조직화 지도로 self-organizing map (SOM) 세계적인 명성을 얻었다. SOM은 그의 이름을 기려 Kohonen 맵이라고도 부른다. 패턴 인식에서 SOM은 군집화에 사용된다. Kohonen은 SOM을 지도 학습으로 확장한 LVQ (learning vector quantization) 학습 알고리즘도 개발하였다. 근래에는 SOM을 웹 문서 검색에 응용한 WEBSOM을 개발하였다 [WebSOM(웹)]. 현재 Academy of Finland의 명예 교수이다.

[WebSOM(웹)] http://websom.hut.fi/websom/.