

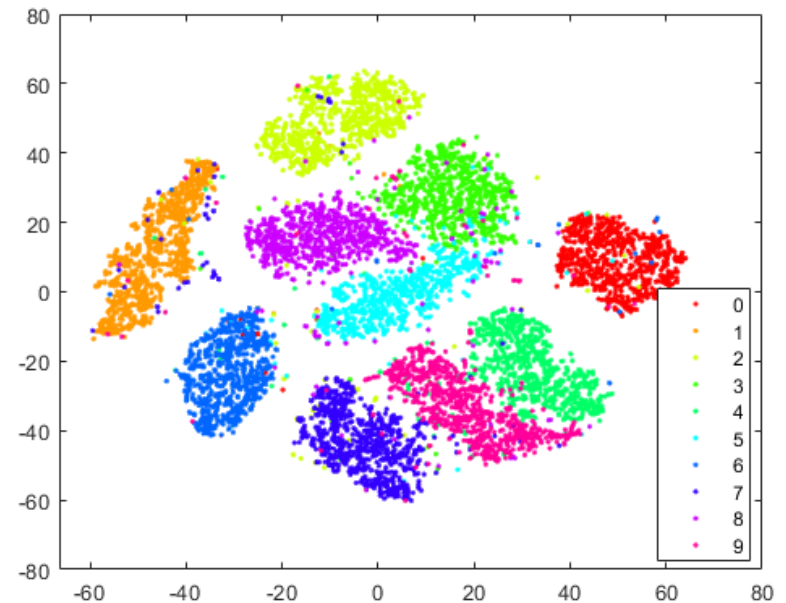
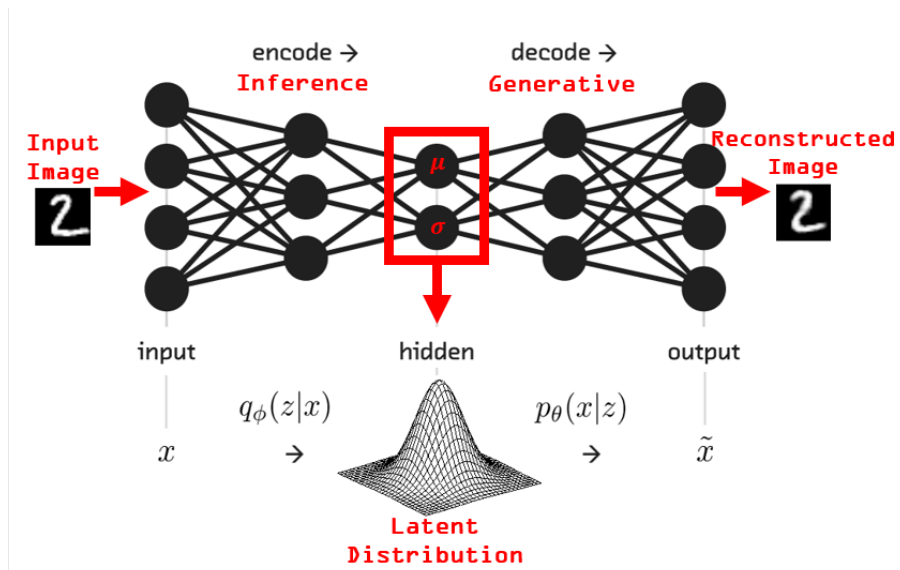
MACHINE 기계 학습 **LEARNING**

오일석 지음

6장. 비지도 학습

PREVIEW

- 지도 학습(supervised learning)과 비지도 학습(unsupervised learning)
 - 지금까지는 훈련집합으로 \mathbb{X} 와 \mathbb{Y} 가 주어지는 지도 학습 supervised learning
 - 6장은 \mathbb{X} 만 주어지는 비지도 학습 unsupervised learning
- 다양한 응용
 - 맞춤 광고, 차원 축소, 데이터 압축, 데이터 가시화, 특징 추출, 생성 모델 구축 등
- 현대 기계 학습에서 더욱 중요해짐



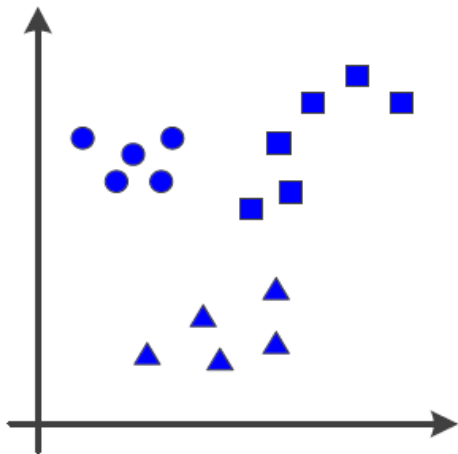
각 절에서 다루는 내용

- 6.1절 비지도 학습을 지도 학습, 준지도 학습과 비교한다.
- 6.2절 비지도 학습의 일반 연산으로 군집화, 밀도 추정, 공간 변환을 소개한다.
- 6.3절 군집화 알고리즘으로 k -평균 알고리즘을 설명한다.
- 6.4절 밀도 추정 방법으로 커널 밀도 추정과 가우시안 혼합을 설명한다.
- 6.5절 기계 학습에서 공간 변환의 중요성을 강조한다.
- 6.6절 선형 인자 모델로서 PCA, ICA, 희소 코딩을 소개한다.
- 6.7절 오토인코더를 소개하고 규제 오토인코더로서 SAE, DAE, CAE를 설명한다.

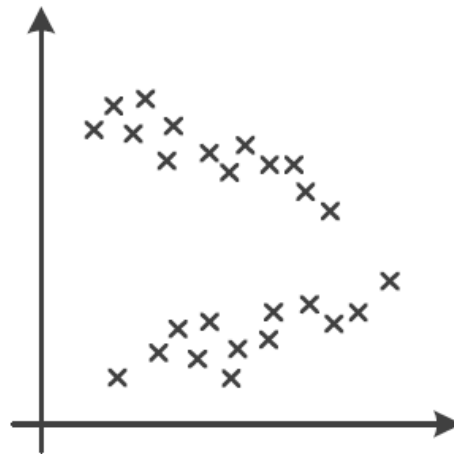
6.1 지도 학습과 비지도 학습, 준지도 학습

■ 세 가지 유형의 학습

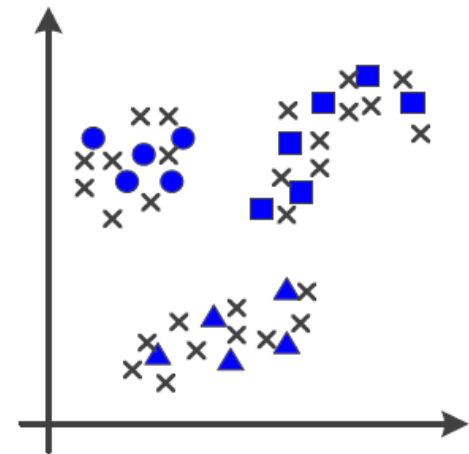
- 지도 학습: 모든 훈련 샘플이 레이블 정보를 가짐
- 비지도 학습: 모든 훈련 샘플이 레이블 정보를 가지지 않음 ← 6장의 주제
- 준지도 학습: 레이블을 가진 샘플과 가지지 않은 샘플이 섞여 있음 ← 7장의 주제



(a) 지도 학습



(b) 비지도 학습



(c) 준지도 학습

그림 6-1 기계 학습의 유형(속이 찬 샘플은 레이블이 있고, x 표시된 샘플은 레이블이 없음)

6.1 지도 학습과 비지도 학습, 준지도 학습

■ 기계 학습이 사용하는 두 종류의 지식

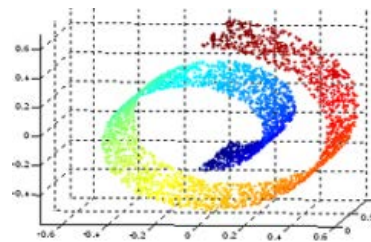
- 훈련집합
- 사전 지식

prior knowledge(세상의 일반적인 규칙)

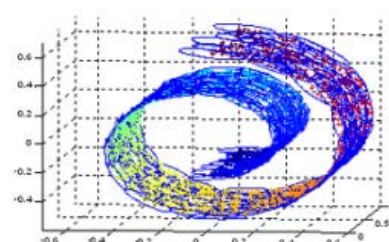
■ 중요한 두 가지 사전 지식

- 매니폴드 가정(manifold hypothesis): 데이터집합은 하나의 매니폴드 또는 여러 개의 매니폴드를 구성하며, 모든 샘플은 매니폴드와 가까운 곳에 있다. 매니폴드와 매니폴드 가정은 6.8.1절에서 자세히 설명한다.
- 매끄러움 가정(smoothness hypothesis): 샘플은 어떤 요인에 의해 변화한다. 예를 들어, 장면과 카메라 위치를 고정한 상태에서 조명을 조금씩 변화하면서 영상을 획득한 경우, 획득된 영상 샘플은 특징 공간에서 위치가 조금씩 바뀔 것이다. 이때 [그림 6-1(b)]와 같이 매끄러운 곡면을 따라 위치가 변한다.

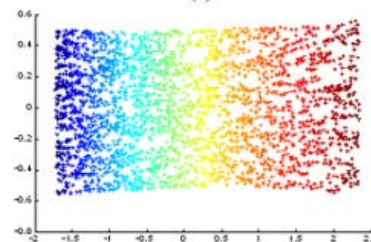
비지도 학습과 준지도 학습은
사전 지식을 더 명시적으로 사용



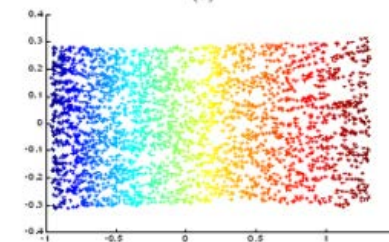
(a)



(b)



(c)



(d)

6.2 비지도 학습

- 6.2.1 비지도 학습의 일반 과업
- 6.2.2 비지도 학습의 응용 과업

6.2.1 비지도 학습의 일반 과업

■ 세 가지 일반 과업

- 군집화: 유사한 샘플을 모아 같은 그룹으로 묶는 일
- 밀도 추정: 데이터로부터 확률분포를 추정하는 일
- 공간 변환: 원래 특징 공간을 저차원 또는 고차원 공간으로 변환하는 일

■ 데이터에 내재한 구조를 잘 파악하여 새로운 정보를 발견해야 함

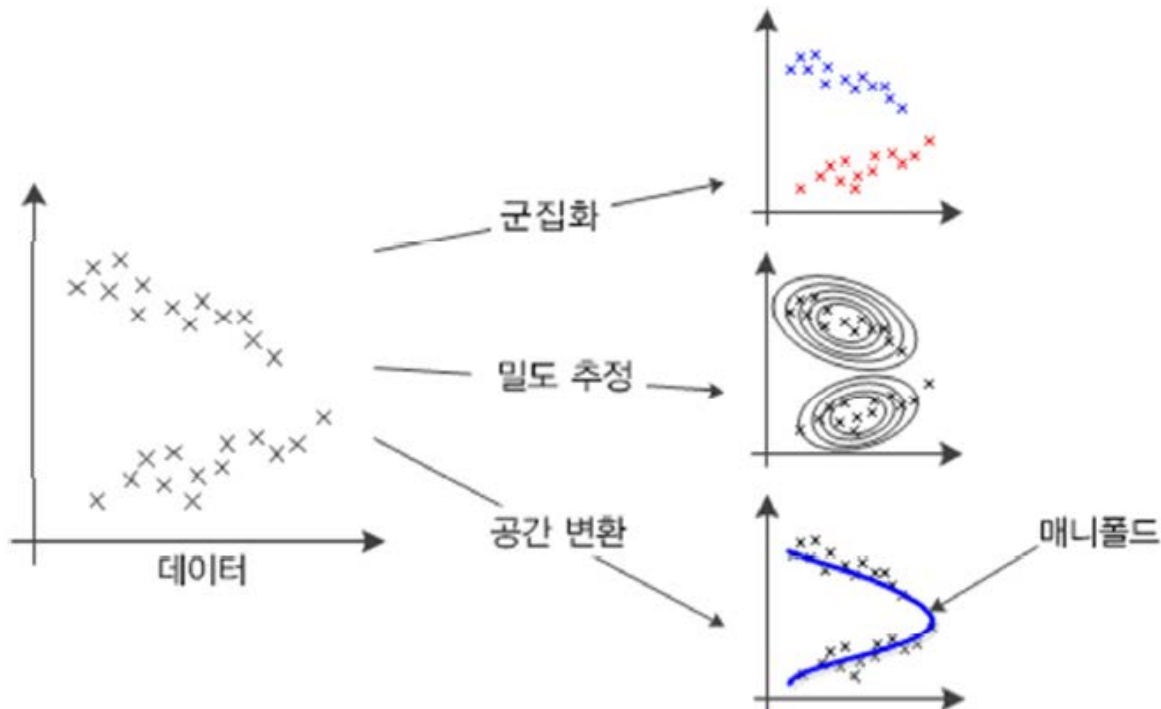


그림 6-2 비지도 학습의 군집화, 밀도 추정, 공간 변환 과업이 발견하는 정보

6.2.2 비지도 학습의 응용 과업

■ 아주 많은 응용(서로 밀접하게 연관)

■ 군집화의 응용

- 맞춤형 광고, 영상 분할, 유전자 데이터 분석, SNS 실시간 검색어 분석하여 사람들의 관심 파악 등

■ 밀도 추정의 응용

- 분류, 생성 모델 구축 등

■ 공간 변환의 응용

- 데이터 가시화, 데이터 압축, 특징 추출(표현 학습) 등



6.3 군집화(clustering)

■ 6.3.1 k -평균(k -means)알고리즘

6.3 군집화(clustering)

■ 군집화 문제

- $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ 에서 식 (6.1)을 만족하는 군집집합 $C = \{c_1, c_2, \dots, c_k\}$ 를 찾아내는 작업

$$\left. \begin{array}{l} c_i \neq \emptyset, i = 1, 2, \dots, k \\ \bigcup_{i=1}^k c_i = \mathbb{X} \\ c_i \cap c_j = \emptyset, i \neq j \end{array} \right\} \quad (6.1)$$

- 군집의 개수 k 는 주어지는 경우와 자동으로 찾아야 하는 경우가 있음
- 군집화를 부류 발견 작업이라 부르기도 함

■ 군집화의 주관성 (최적의 k 개수?)

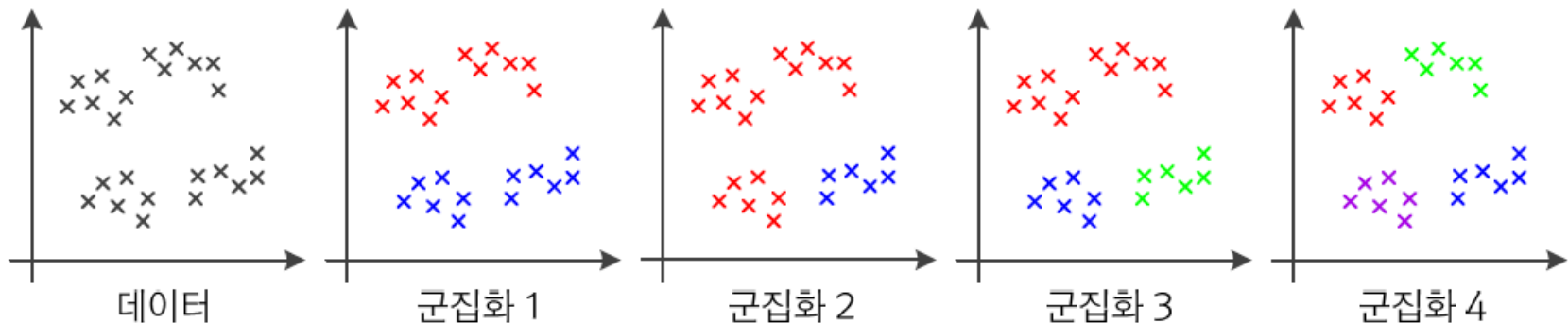
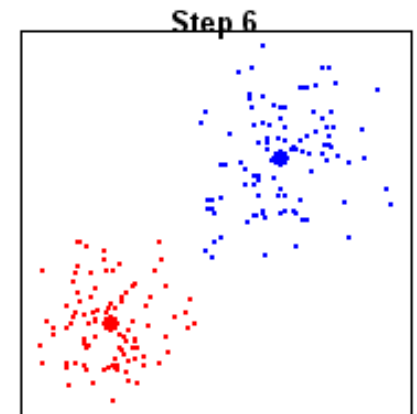
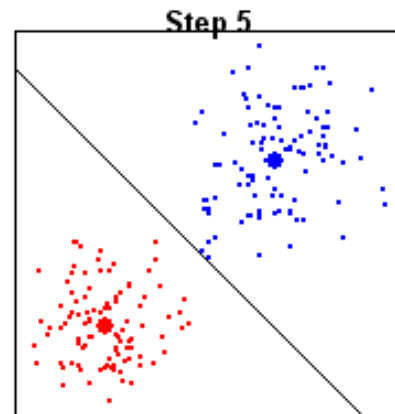
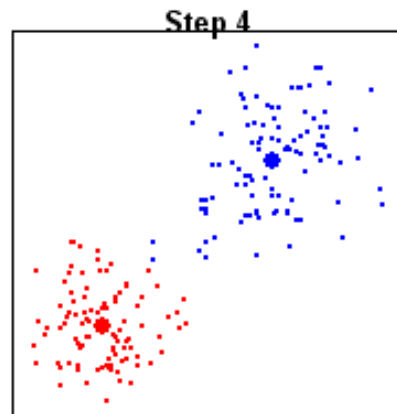
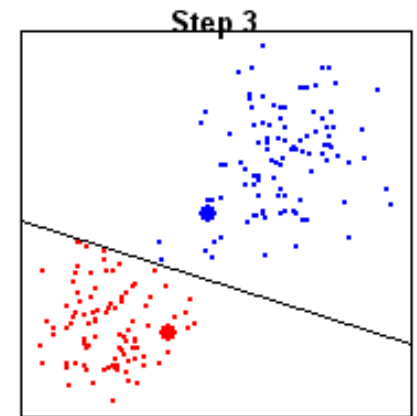
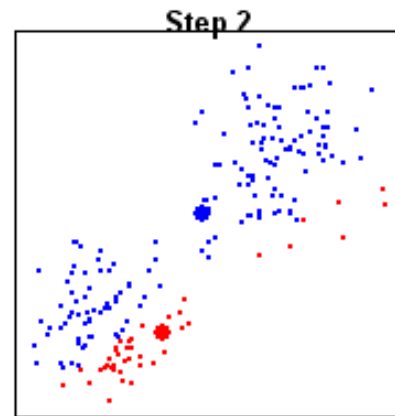
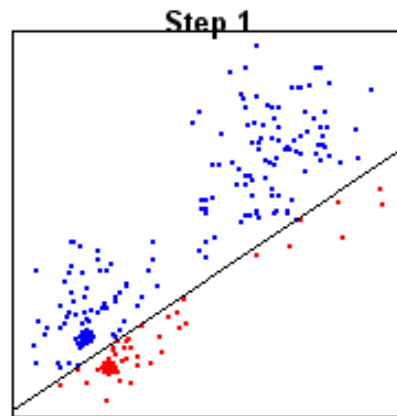


그림 6-3 군집화의 주관성

6.3.1 k -평균 알고리즘

■ k -평균 알고리즘의 특성

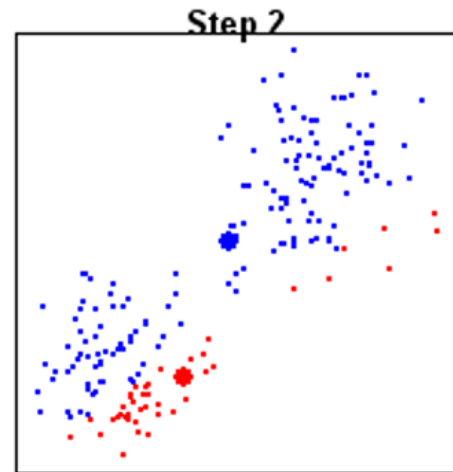
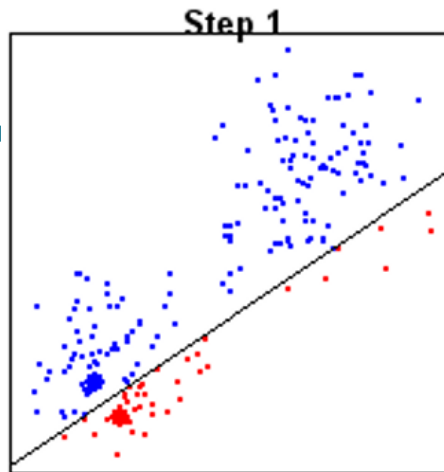
- 군집 개수(k)와 군집 중심의 초기 위치(z_1, z_2, \dots, z_k)가 주어질 때,
 - 1. 각 샘플 별로 가까운 군집으로 할당
 - 2. 군집 중심의 위치는 군집의 할당된 샘플 평균으로 갱신
 - 3. 1과 2를 반복



6.3.1 k -평균 알고리즘

■ k -평균 알고리즘의 특성

- 원리 단순하지만 성능이 좋아 인기 좋음
- 직관적으로 이해하기 쉽고 구현 쉬움
- 군집 개수 k 를 알려줘야 함



알고리즘 6-1 k -평균

입력: 훈련집합 $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, 군집의 개수 k

출력: 군집집합 $C = \{c_1, c_2, \dots, c_k\}$

```
1   $k$ 개의 군집 중심  $Z = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_k\}$ 를 초기화한다.
2  while (true)
3      for ( $i=1$  to  $n$ )
4           $\mathbf{x}_i$ 를 가장 가까운 군집 중심에 배정한다.
5          if (라인 3~4에서 이루어진 배정이 이전 루프에서의 배정과 같으면) break
6      for ( $j=1$  to  $k$ )
7           $\mathbf{z}_j$ 에 배정된 샘플의 평균으로  $\mathbf{z}_j$ 를 대체한다.
8  for ( $j=1$  to  $k$ )
9       $\mathbf{z}_j$ 에 배정된 샘플을  $c_j$ 에 대입한다.
```

6.3.1 k -평균 알고리즘

■ k -평균과 k -medoids

- k -평균은 [알고리즘 6-1]의 라인 7에서 샘플의 평균으로 군집 중심을 갱신
- k -medoids는 대표를 뽑아 뽑힌 대표로 군집 중심을 갱신(k -평균에 비해 잡음에 둔감)



그림 6-4 k -평균과 k -medoids가 군집 중심을 갱신하는 과정

■ 최적화 문제로 해석

- k -평균은 식 (6.2)의 목적함수를 최소화하는 알고리즘
- 행렬 \mathbf{A} 는 군집 배정 정보를 나타내는 $k \times n$ 행렬(i 번째 샘플이 j 번째 군집에 배정되었다면 a_{ji} 는 1, 그렇지 않으면 0)

$$J(\mathbf{Z}, \mathbf{A}) = \sum_{i=1}^n \sum_{j=1}^k a_{ji} \text{dist}(\mathbf{x}_i, \mathbf{z}_j) \quad (6.2)$$

6.3.1 k -평균 알고리즘

예제 6-1 k -평균의 동작

[그림 6-5]는 훈련집합이 7개의 샘플을 가진 $n=7$ 인 예를 보여 준다. 좌표는 다음과 같다.

$$\mathbf{x}_1 = \begin{pmatrix} 18 \\ 5 \end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix} 20 \\ 9 \end{pmatrix}, \mathbf{x}_3 = \begin{pmatrix} 20 \\ 14 \end{pmatrix}, \mathbf{x}_4 = \begin{pmatrix} 20 \\ 17 \end{pmatrix}, \mathbf{x}_5 = \begin{pmatrix} 5 \\ 15 \end{pmatrix}, \mathbf{x}_6 = \begin{pmatrix} 9 \\ 15 \end{pmatrix}, \mathbf{x}_7 = \begin{pmatrix} 6 \\ 20 \end{pmatrix}$$

군집의 개수 $k=3$ 이라 하자. 맨 왼쪽 그림은 초기 군집 중심을 보여 준다. [알고리즘 6-1]의 라인 3~4는 7개 샘플을 아래와 같이 배정할 것이다.

$$\{\mathbf{x}_1\} \text{은 } \mathbf{z}_1, \{\mathbf{x}_2\} \text{은 } \mathbf{z}_2, \{\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7\} \text{은 } \mathbf{z}_3$$

이 배정을 행렬 \mathbf{A} 로 표현하면 다음과 같다.

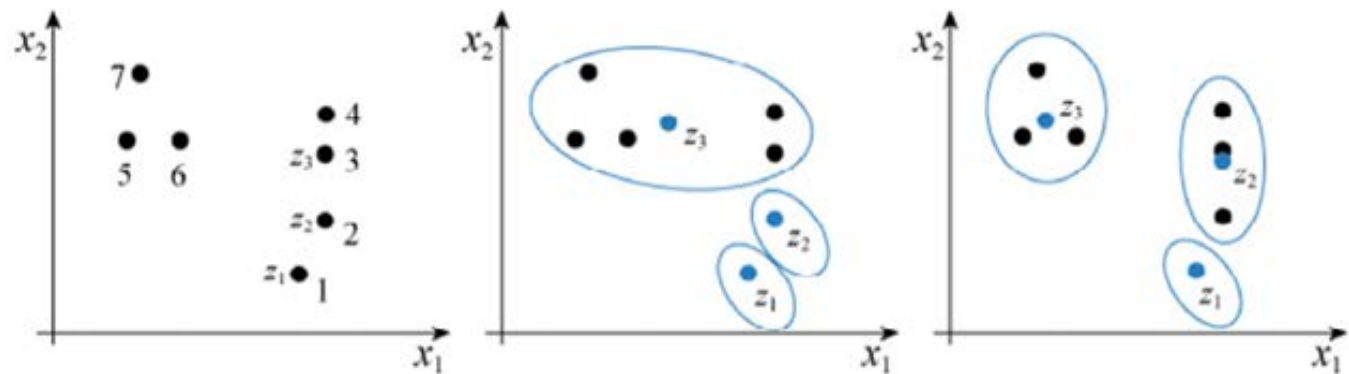
$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$


그림 6-5 k -평균의 동작 예제

6.3.1 k -평균 알고리즘

[그림 6-5]의 가운데 그림은 새로 계산한 군집 중심이다. $\mathbf{z}_1 = (18, 5)^T$, $\mathbf{z}_2 = (20, 9)^T$, $\mathbf{z}_3 = (12, 16.2)^T$ 이고, 식 (6.2)에 대입하면 $J = 244.80$ 이 된다. 이때 거리함수 dist로 식 (1.7)의 유클리디언 거리를 사용한다.

두 번째 루프를 실행하면 행렬 \mathbf{A} 는 아래와 같이 바뀐다. 군집 중심은 $\mathbf{z}_1 = (18, 5)^T$, $\mathbf{z}_2 = (20, 13.333)^T$, $\mathbf{z}_3 = (6.667, 16.667)^T$ 이다. 이것을 식 (6.2)에 대입하면 $J = 58.00$ 이 된다. [그림 6-5]의 맨 오른쪽 그림은 두 번째 루프 수행 후의 상황이다.

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

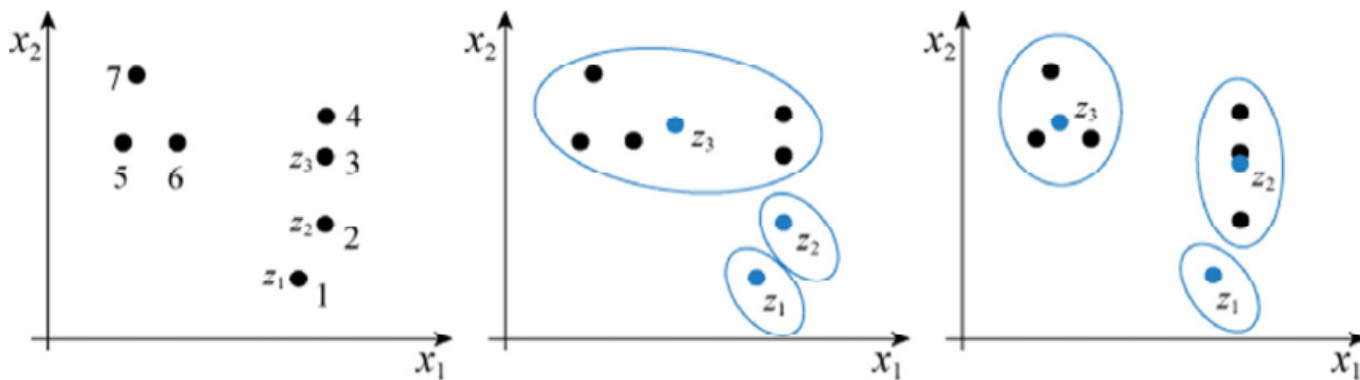


그림 6-5 k -평균의 동작 예제

6.3.1 k -평균 알고리즘

■ EM(Expectation Maximization)기초

- k -평균에서 훈련집합 \mathbb{X} 와 군집집합 C (행렬 A)는 각각 입력단과 출력단에서 관찰 가능
- 중간 단계의 입시 변수 Z (입출력단에서 보이지 않기 때문에 은닉변수라^{latent variable} 부름)
- k -평균은 Z 의 추정(E 단계)과 A 의 추정(M 단계)을 번갈아 가면 수행하는 EM 알고리즘

알고리즘 6-1 k -평균

입력: 훈련집합 $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, 군집의 개수 k

출력: 군집집합 $C = \{c_1, c_2, \dots, c_k\}$

```
1   $k$ 개의 군집 중심  $Z = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_k\}$ 를 초기화한다.
2  while (true)
3    for ( $i=1$  to  $n$ )
4       $\mathbf{x}_i$ 를 가장 가까운 군집 중심에 배정한다.
5    if (라인 3~4에서 이루어진 배정이 이전 루프에서의 배정과 같으면) break
6    for ( $j=1$  to  $k$ )
7       $\mathbf{z}_j$ 에 배정된 샘플의 평균으로  $\mathbf{z}_j$ 를 대체한다.
8  for ( $j=1$  to  $k$ )
9     $\mathbf{z}_j$ 에 배정된 샘플을  $c_j$ 에 대입한다.
```

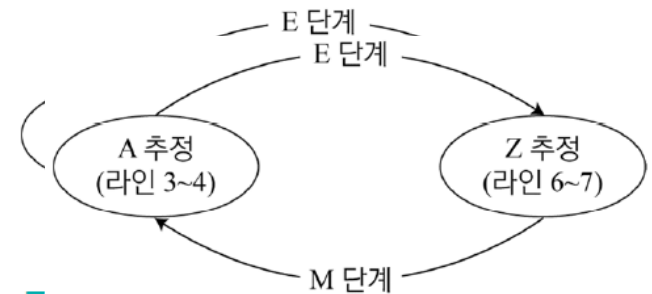
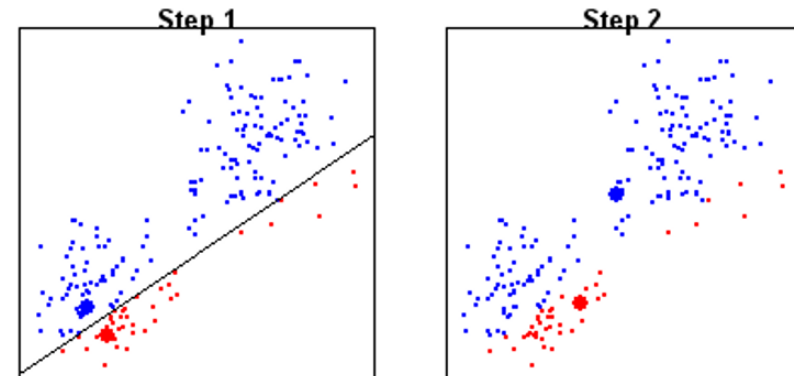


그림 6-6 k -평균을 EM 알고리즘으로 해석



6.4 밀도 추정

- 6.4.1 커널 밀도 추정
- 6.4.2 가우시안 혼합
- 6.4.3 EM 알고리즘

■ 밀도 추정 문제

- 어떤 점 \mathbf{x} 에서 데이터가 발생할 확률, 즉 확률 분포 $P(\mathbf{x})$ 를 구하는 문제
- 예를 들어, 그림 6-8에서 $P(\mathbf{x}_1) > P(\mathbf{x}_2) > P(\mathbf{x}_3)$

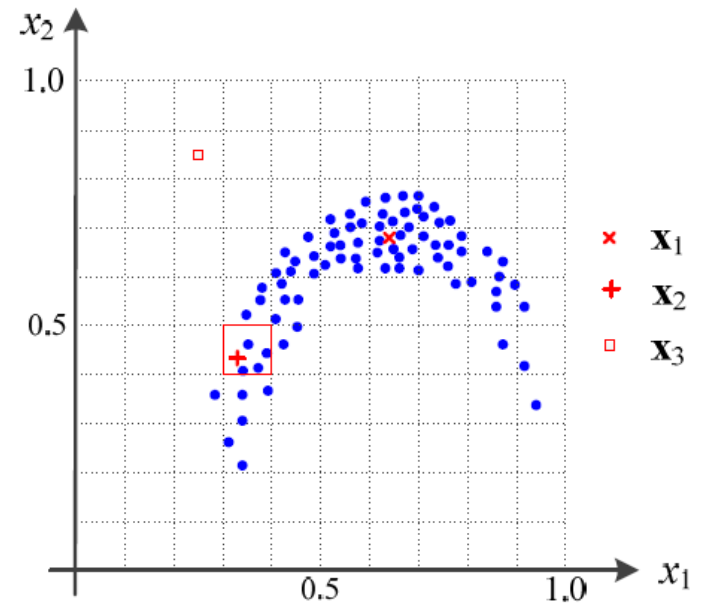


그림 6-8 밀도 추정 문제

6.4.1 커널(kernel) 밀도 추정

■ 히스토그램 방법

- 특징 공간을 칸의 집합으로 분할한 다음, 칸에 있는 샘플의 빈도를 세어 식 (6.7)로 추정

- 예, $P(\mathbf{x} = +) = \frac{4}{80} = 0.05$

$$P(\mathbf{x}) = \frac{\text{bin}(\mathbf{x})}{n} \quad (6.7)$$

- 여러 문제점

- 매끄럽지 못하고 계단 모양을 띠는 확률밀도함수가 됨
- 칸의 크기와 위치에 민감함

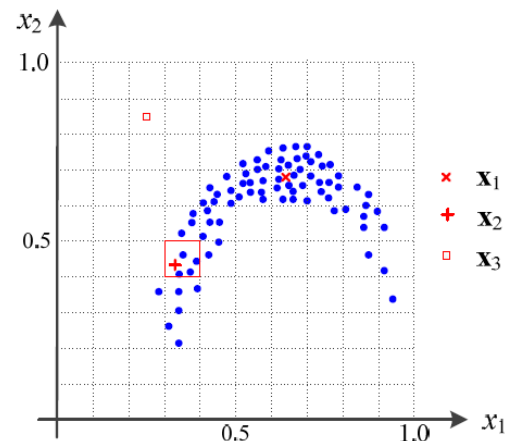


그림 6-8 밀도 추정 문제

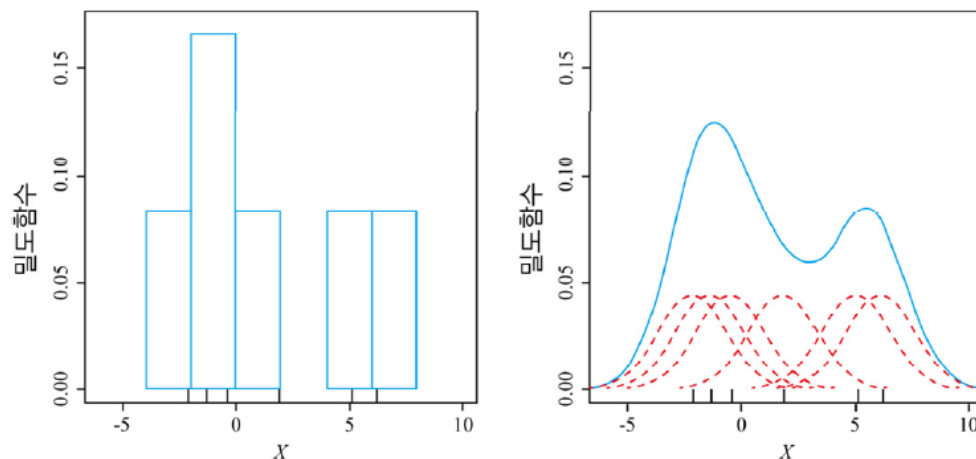


그림 6-10 히스토그램 방법(왼쪽)과 커널 밀도 추정법(오른쪽)의 비교

6.4.1 커널 밀도 추정

■ 커널 밀도 추정법

- 커널(kernel): n 차원공간을 n' 차원 공간으로 사상하는 함수
- 점 \mathbf{x} 에 [그림 6-9]가 예시하는 커널을 씌우고 커널 안에 있는 샘플의 가중 합을 이용함
- 대역폭 h 의 크기가 중요

$$P_h(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n K_h(\mathbf{x} - \mathbf{x}_i) = \frac{1}{nh^d} \sum_{i=1}^n K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \quad (6.8)$$

여기서 $K_h(\mathbf{x}) = \frac{1}{h^d} K\left(\frac{\mathbf{x}}{h}\right)$

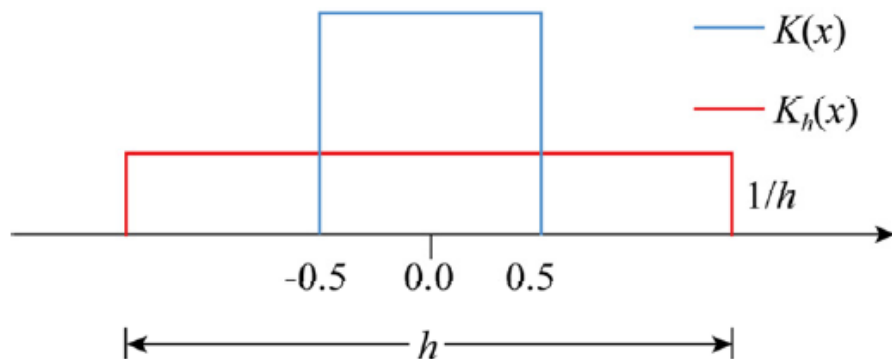


그림 6-9 표준커널함수 K 와 크기 변환된 커널함수 K_h

6.4.1 커널 밀도 추정

- 히스토그램 방법과 커널 밀도 추정법의 비교
 - 커널 밀도 추정법은 매끄러운 확률밀도함수를 추정함

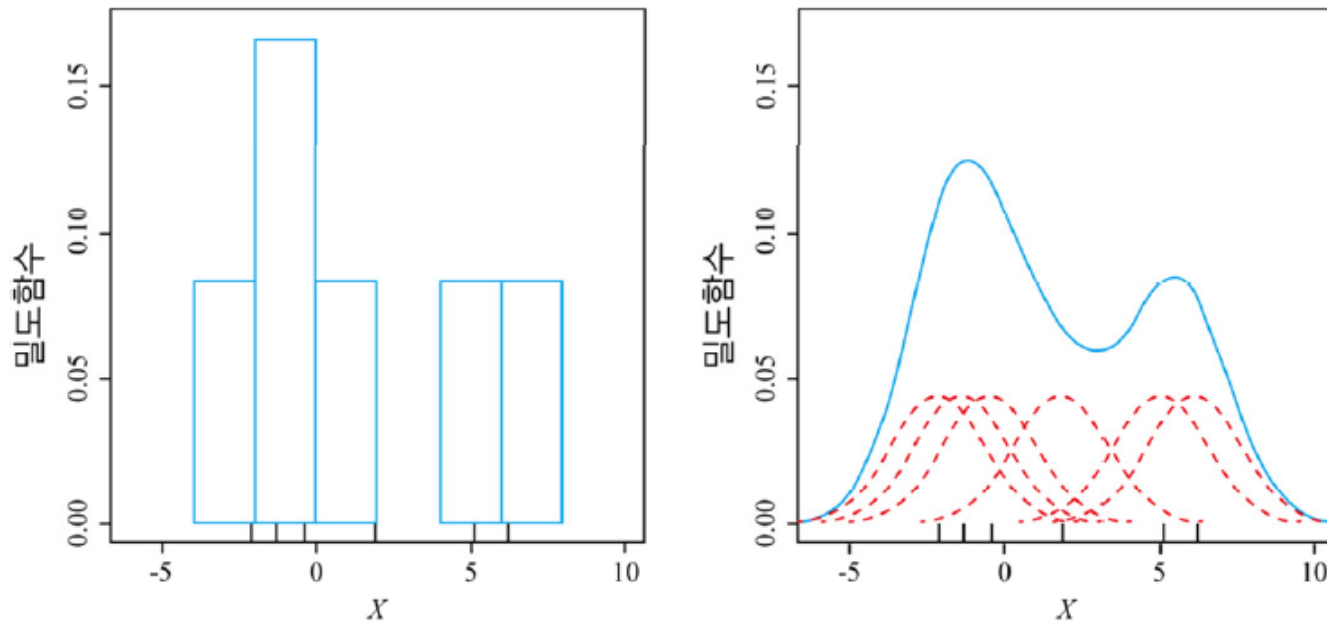


그림 6-10 히스토그램 방법(왼쪽)과 커널 밀도 추정법(오른쪽)의 비교

6.4.1 커널 밀도 추정

■ 커널 밀도 추정법에서 대역폭 h 의 중요성

- h 가 너무 작으면(빨강) 뾰족뾰족한 모양

h 가 너무 크면(녹색) 뭉개짐

→ 적절하게 설정해야 함(검정)

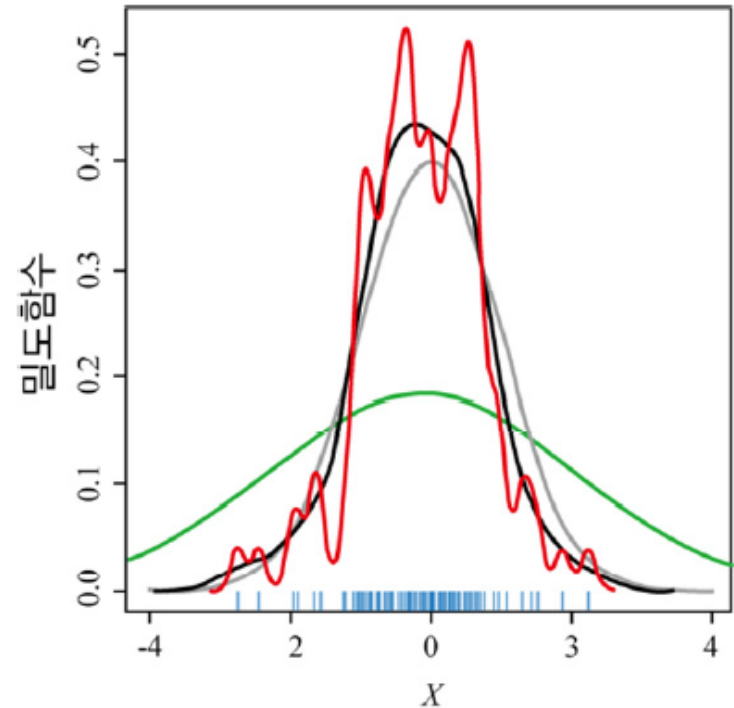


그림 6-11 대역폭이 확률밀도함수 추정에 미치는 영향

■ 커널 밀도 추정 기법의 근본적 문제점

- 샘플을 모두 저장하고 있어야 하는 메모리 기반 방법(새로운 샘플이 주어질 때마다 식 (6.8)을 처음부터 다시 계산)
- 데이터 희소성(차원의 저주)

→ 데이터가 낮은 차원인 경우로 국한하여 활용

6.4.2 가우시안 혼합(Gaussian mixture)

■ 가우시안을 이용한 방법(모수적 방법: parametric method)

- 데이터가 가우시안 분포를 따른다고 가정하고 평균 벡터 μ 와 공분산 행렬 Σ 를 추정함

$$P(\mathbf{x}) = N(\mathbf{x}; \mu, \Sigma) = \frac{1}{\sqrt{|\Sigma|}\sqrt{(2\pi)^d}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right)$$

이때 $\mu = \frac{1}{n} \sum_{i=1,n} \mathbf{x}_i, \Sigma = \frac{1}{n} \sum_{i=1,n} (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T$

(6.9)

■ 대부분 데이터가 하나의 가우시안으로 불충분([그림 6-12]의 오른쪽)

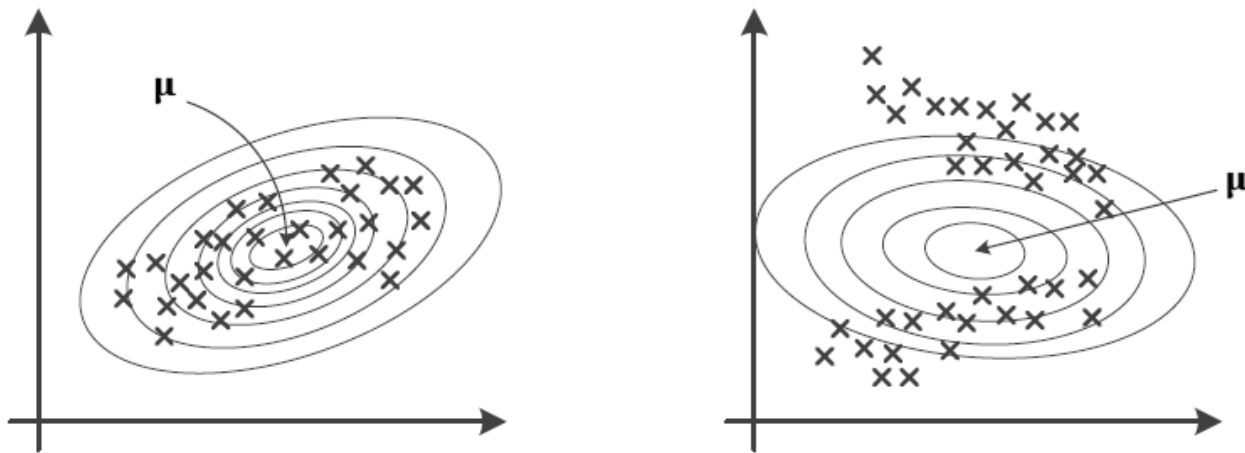
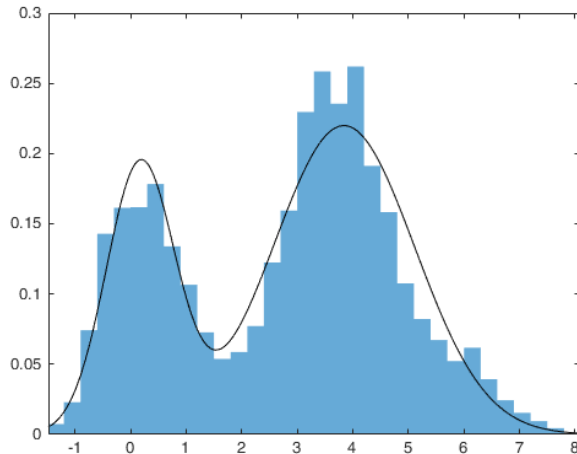


그림 6-12 하나의 가우시안으로 밀도 추정

6.4.2 가우시안 혼합

■ 가우시안 혼합

- [그림 6-13]은 2개의 가우시안을 사용한 예



가우시안 혼합 모델($k=2$)

■ k 개의 가우시안으로 일반화하면,

- 확률분포 $P(\mathbf{x})$ 는 k 개 가우시안의 선형 결합으로 표현(식 (6.10))

$$P(\mathbf{x}) = \sum_{j=1}^k \pi_j N(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \quad (6.10)$$

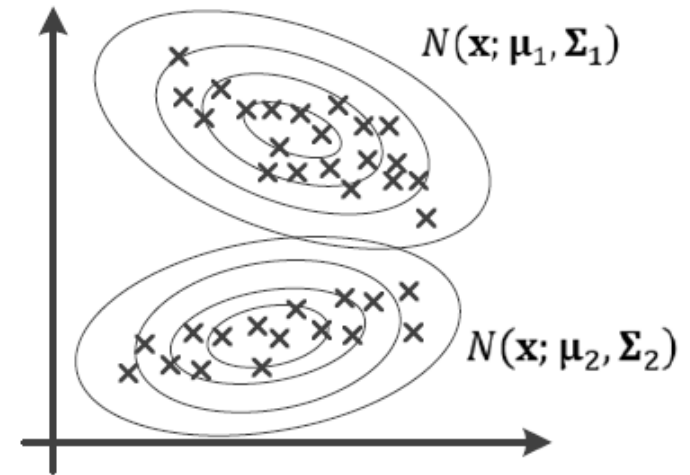


그림 6-13 가우시안 혼합으로 밀도 추정

$$P(\mathbf{x}) = \frac{21}{37} N(\mathbf{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + \frac{16}{37} N(\mathbf{x}; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$

6.4.2 가우시안 혼합

- 주어진 데이터와 추정해야 할 매개변수를 정리하면,

주어진 데이터: 훈련집합 $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, 가우시안의 개수 k

추정해야 할 매개변수집합: $\Theta = \{\boldsymbol{\pi} = (\pi_1, \pi_2, \dots, \pi_k), (\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1), (\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2), \dots, (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\}$

- 최대 우도를 이용한 최적화 문제로 공식화

$$P(\mathbb{X}|\Theta) = \prod_{i=1}^n P(\mathbf{x}_i|\Theta) = \prod_{i=1}^n \left(\sum_{j=1}^k \pi_j N(\mathbf{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \right) \quad (6.11)$$

$$\log P(\mathbb{X}|\Theta) = \sum_{i=1}^n \log \left(\sum_{j=1}^k \pi_j N(\mathbf{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \right) \quad (6.12)$$

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmax}} \log P(\mathbb{X}|\Theta) \quad (6.13)$$

6.4.3 EM 알고리즘

■ EM 알고리즘을 이용한 식 (6.13)의 풀이

- θ 를 모르므로 난수로 설정하고 출발([그림 6-14]의 예시)

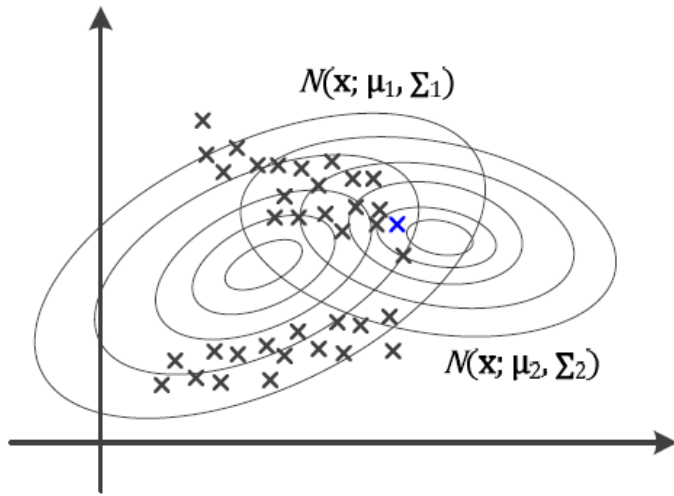


그림 6-14 샘플의 소속 확률을 어떻게 추정할 것인가

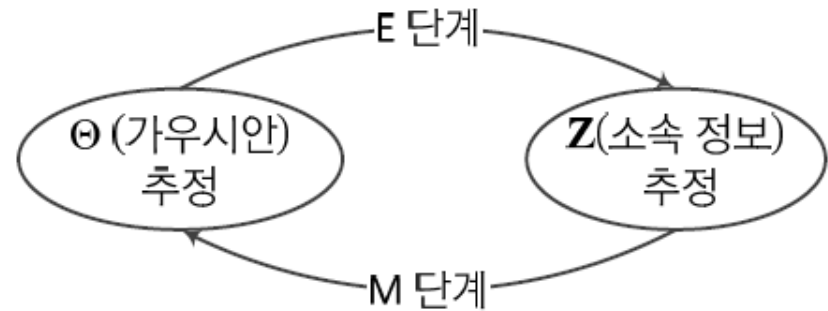


그림 6-15 가우시안 혼합을 위한 EM 알고리즘

- 가우시안으로 샘플의 소속 정보 개선(E단계) → 샘플의 소속 정보로 가우시안 개선(M단계) → 가우시안으로 샘플의 소속 정보 개선(E단계) → 샘플의 소속 정보로 가우시안 개선(M단계) → ([그림 6-15])

6.4.2 가우시안 혼합

■ 가우시안 혼합 알고리즘

- Expectation, Maximization(EM)을 반복

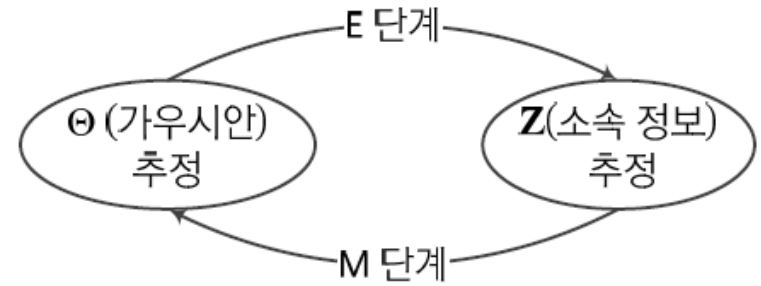
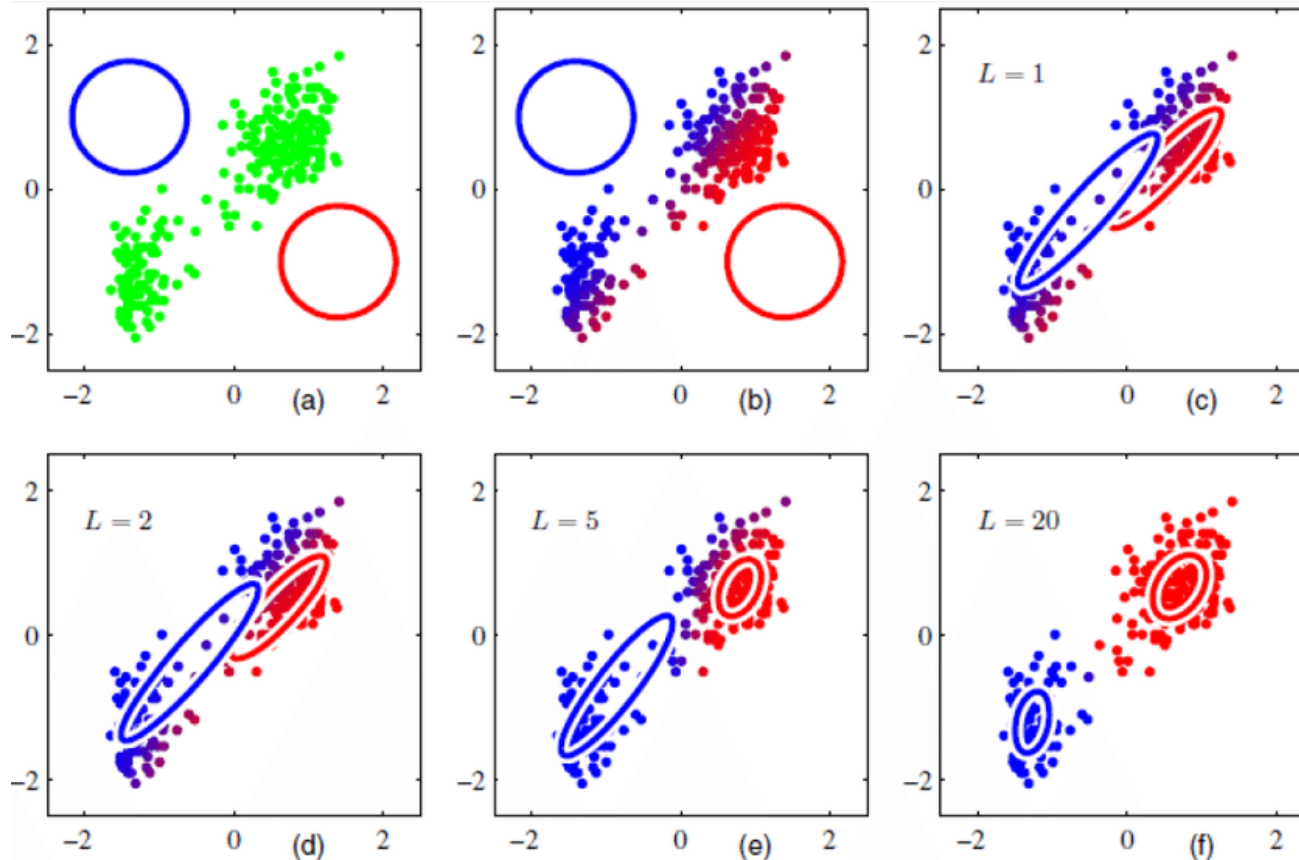


그림 6-15 가우시안 혼합을 위한 EM 알고리즘



6.4.3 EM 알고리즘

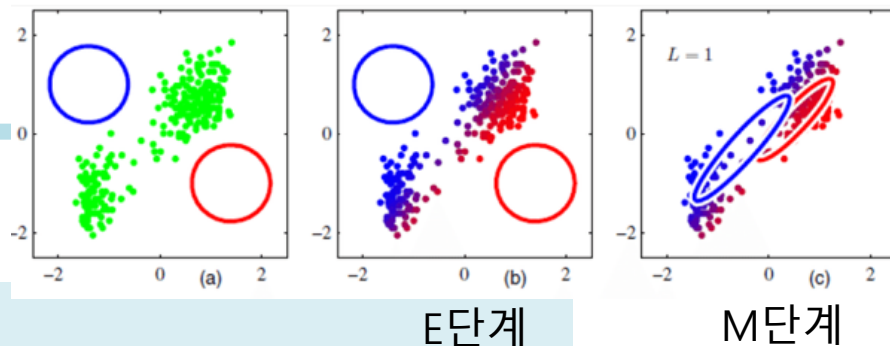
■ 가우시안 혼합을 위한 EM 알고리즘

알고리즘 6-4 가우시안 혼합을 위한 EM 알고리즘

입력: 훈련집합 $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, 가우시안의 개수 k

출력: 최적의 가우시안과 혼합 계수 $\Theta = \{\boldsymbol{\pi} = (\pi_1, \pi_2, \dots, \pi_k), (\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1), (\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2), \dots, (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\}$

- 1 Θ 를 초기화한다.
- 2 while (!멈춤조건)
- 3 Θ 를 이용하여 소속확률 행렬 \mathbf{Z} 를 추정한다. // E단계
- 4 \mathbf{Z} 를 이용하여 Θ 를 추정한다. // M단계



- 라인 3과 라인 4를 위한 수식
- z_{ji} 는 \mathbf{x}_i 가 j 번째 가우시안에 속할 확률

$$z_{ji} = \frac{\pi_j N(\mathbf{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{q=1}^k \pi_q N(\mathbf{x}_i; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)} \quad (6.14)$$

$$\left. \begin{aligned} \boldsymbol{\mu}_j &= \frac{1}{n_j} \sum_{i=1}^n z_{ji} \mathbf{x}_i \\ \boldsymbol{\Sigma}_j &= \frac{1}{n_j} \sum_{i=1}^n z_{ji} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^T \\ \pi_j &= \frac{n_j}{n} \\ \text{이때 } n_j &= \sum_{i=1}^n z_{ji} \end{aligned} \right\} \quad (6.15)$$

6.5 공간 변환의 이해

■ 간단한 상황 예시

- 2개 군집을 가진 [그림 6-16]의 2차원 특징 공간을 극좌표 공간으로 변환하면 1차원만으로 2개 군집 표현 가능

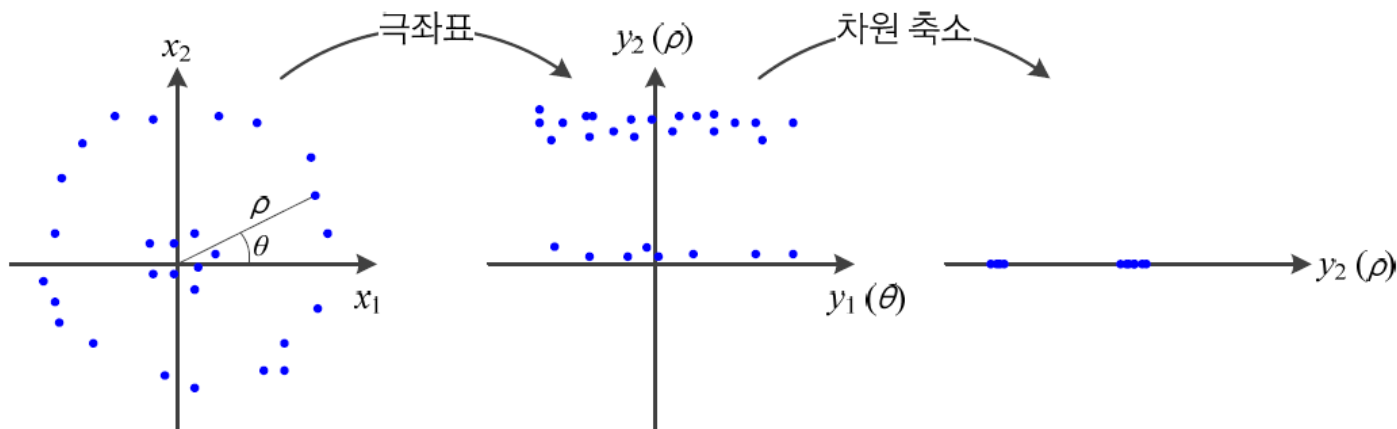


그림 6-16 공간 변환의 예

- 실제 문제에서는 비지도 학습을 이용하여 최적의 공간 변환을 자동으로 알아내야 함

6.5 공간 변환의 이해

■ 인코딩과 디코딩

- 원래 공간을 다른 공간으로 변환하는 인코딩 과정(f), 변환 공간을 원래 공간으로 역변환하는 디코딩 과정(g)

$$\hat{\mathbf{x}} = g(f(\mathbf{x})) \quad (6.16)$$

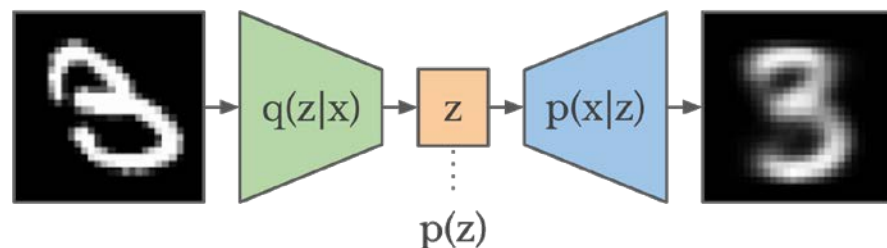
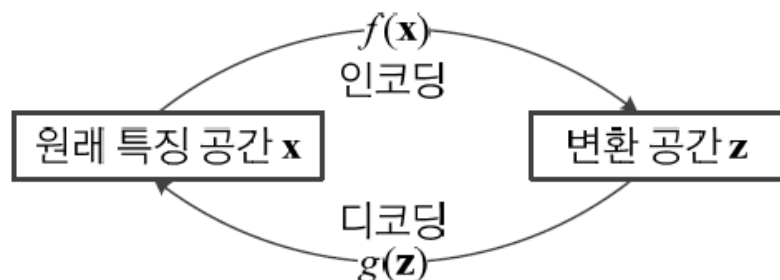


그림 6-17 공간 변환과 역변환

- 예) 데이터 압축의 경우, 역변환으로 얻은 $\hat{\mathbf{x}}$ 은 원래 신호 \mathbf{x} 와 가급적 같아야 함
- 예) 데이터 가시화에서는 2차원 또는 3차원의 \mathbf{z} 공간으로 변환. 디코딩은 불필요

6.6 선형 인자 모델(linear factor model)

- 6.6.1 주성분 분석(PCA, principle component alaysis)
- 6.6.2 독립 성분 분석(ICA, independent component analysis)
- 6.6.3 희소 코딩(SC, sparse coding)

6.6 선형 인자 모델

■ 선형 인자 모델(linear factor model)

- 선형 연산을 이용한 공간 변환 기법
- 선형 연산을 사용하므로 행렬 곱으로 인코딩(식 (6.17))과 디코딩(식 (6.18)) 과정을 표현

$$f: \mathbf{z} = \mathbf{W}_{enc}\mathbf{x} + \boldsymbol{\alpha}_{enc} \quad (6.17)$$

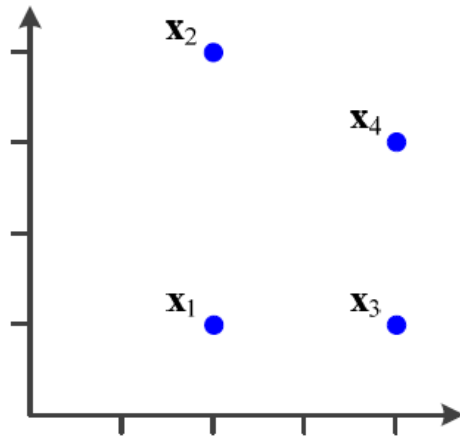
$$g: \mathbf{x} = \mathbf{W}_{dec}\mathbf{z} + \boldsymbol{\alpha}_{dec} \quad (6.18)$$

- $\boldsymbol{\alpha}$ 는 데이터를 원점으로 이동하거나 잡음을 추가하는 등의 역할
- 인자 \mathbf{z} 와 추가 항 $\boldsymbol{\alpha}$ 에 따라 여러 가지 모델이 존재
 - \mathbf{z} 에 확률 개념이 없고 $\boldsymbol{\alpha}$ 를 생략하면 PCA(6.6.1절) – 관찰 벡터 \mathbf{x} 와 인자 \mathbf{z} 는 결정론적인 (deterministic) 1:1 매핑 관계
 - \mathbf{z} 와 $\boldsymbol{\alpha}$ 가 가우시안 분포를 따른다고 가정하면 확률 PCA^{probabilistic PCA}
 - \mathbf{z} 가 비가우시안 분포를 따른다고 가정하는 ICA(6.6.2절)

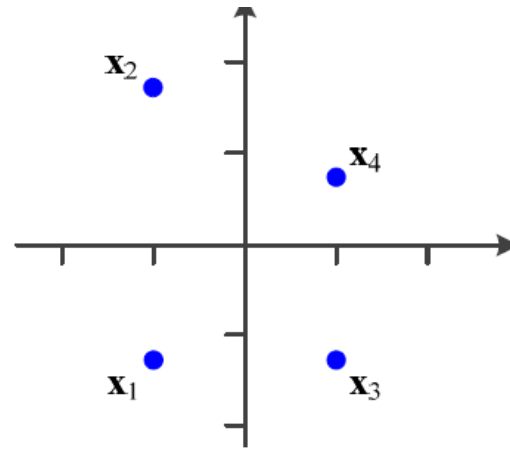
6.6.1 주성분 분석(PCA, principle component analysis)

- 데이터를 원점 중심으로 옮기는 전처리를 먼저 수행

$$\left. \begin{array}{l} \mathbf{x}_i = \mathbf{x}_i - \boldsymbol{\mu}, \quad i = 1, 2, \dots, n \\ \text{이때 } \boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \end{array} \right\} \quad (6.19)$$



(a) 원래 훈련집합 \mathbb{X}



(b) \mathbb{X} 에 식 (6.19)를 적용한 이후

그림 6-18 \mathbb{X} 의 평균을 0으로 변환

6.6.1 주성분 분석(PCA, principle component analysis)

■ 주성분 분석이 사용하는 변환식

- 일반적인 선형 변환식인 식 (6.17)에서 \mathbf{z} 에 확률 개념이 없고 $\boldsymbol{\alpha}$ 를 생략하면 주성분 분석
- 변환 행렬 \mathbf{W} 는 $d * q$ 로서 주성분 분석은 d 차원의 \mathbf{x} 를 q 차원의 \mathbf{z} 로 변환 ($q < d$)
 - \mathbf{W} 의 j 번째 열 벡터와의 내적 $\mathbf{u}_j^T \mathbf{x}$ 는 \mathbf{x} 를 \mathbf{u}_j 가 가리키는 축으로 투영(projection)

$$\left. \begin{aligned} \mathbf{z} &= \mathbf{W}^T \mathbf{x} \\ \text{이때 } \mathbf{W} &= (\mathbf{u}_1 \ \mathbf{u}_2 \ \cdots \ \mathbf{u}_q) \text{이고, } \mathbf{u}_j = (u_{1j}, u_{2j}, \cdots, u_{dj})^T \end{aligned} \right\} \quad (6.20)$$

- 예, 2차원을 1차원으로 변환하는 상황($d = 2, q = 1$)

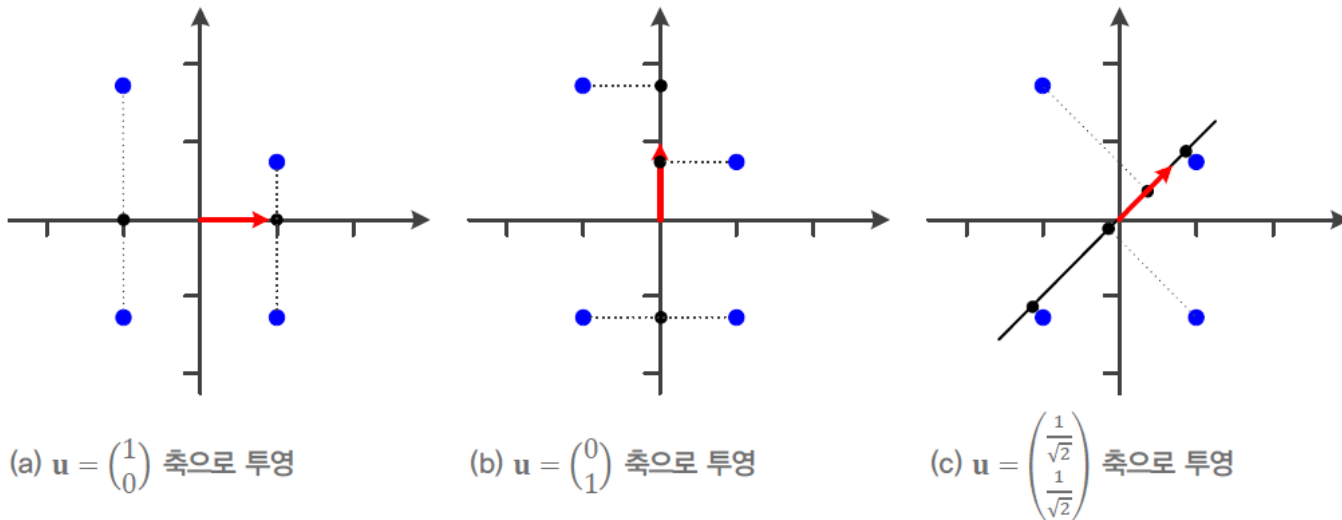


그림 6-19 투영에 의해 2차원을 1차원으로 변환

6.6.1 주성분 분석

■ 주성분 분석의 목적

- 손실을 최소화하면서 저차원으로 변환하는 것
 - [그림 6-19]에서 정보 손실 예
 - [그림 6-19(a)]는 x_1 과 x_2 쌍, x_3 과 x_4 쌍이 같은 점으로 변환되는 정보 손실
 - [그림 6-19(b)]는 x_1 과 x_3 쌍이 같은 점으로 변환되는 정보 손실
 - [그림 6-19(c)]는 4개 점이 모두 다른 점으로 변환되어 정보 손실이 가장 적음
- 주성분 분석은 변환된 훈련집합 $\mathbb{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n\}$ 의 **분산**이 클수록 정보 손실이 적다고

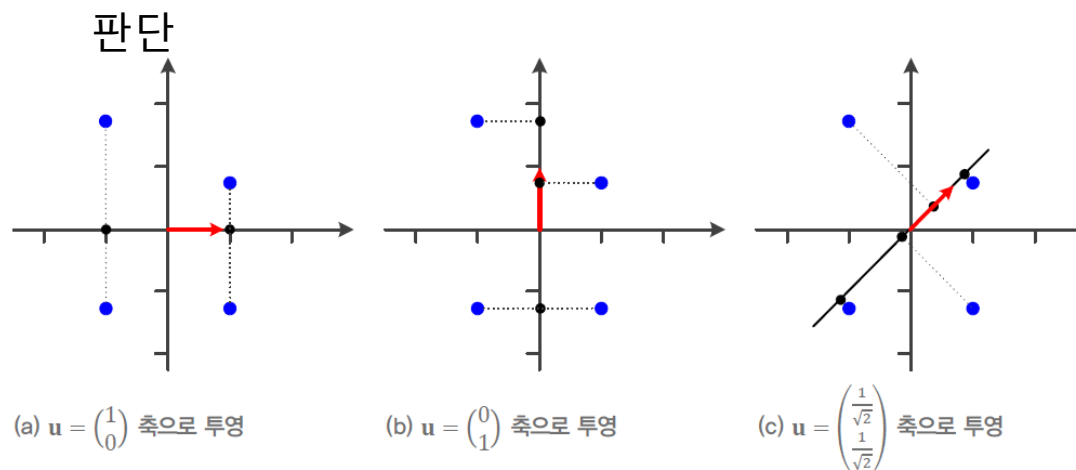
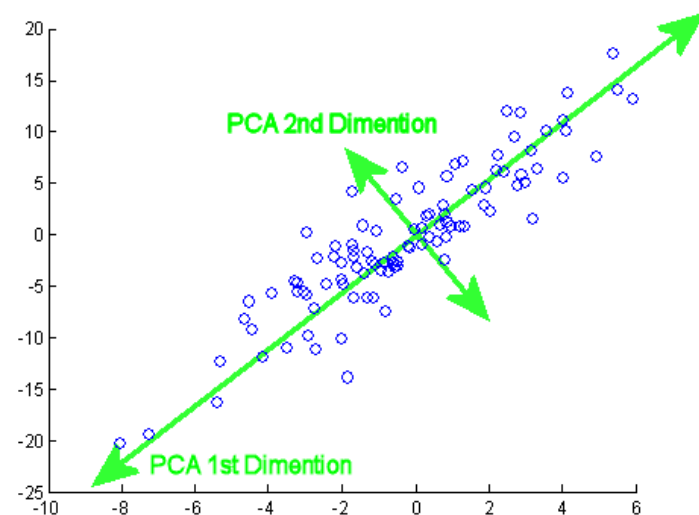


그림 6-19 투영에 의해 2차원을 1차원으로 변환



6.6.1 주성분 분석

예제 6-2 [그림 6-19]의 세 가지 경우의 분산

[그림 6-18(a)]의 훈련집합에 식 (6.19)를 적용하기 전과 후는 다음과 같다.

$$\mathbf{x}_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix} 2 \\ 4 \end{pmatrix}, \mathbf{x}_3 = \begin{pmatrix} 4 \\ 1 \end{pmatrix}, \mathbf{x}_4 = \begin{pmatrix} 4 \\ 3 \end{pmatrix} \Rightarrow \mathbf{x}_1 = \begin{pmatrix} -1 \\ -1.25 \end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix} -1 \\ 1.75 \end{pmatrix}, \mathbf{x}_3 = \begin{pmatrix} 1 \\ -1.25 \end{pmatrix}, \mathbf{x}_4 = \begin{pmatrix} 1 \\ 0.75 \end{pmatrix}$$

[그림 6-19(a)]의 $\mathbf{u} = (1 \ 0)^T$ 축으로 투영된 점은 다음과 같다. $z_1 \sim z_4$ 의 분산은 1.00이다.

$$z_1 = (1 \ 0) \begin{pmatrix} -1 \\ -1.25 \end{pmatrix} = -1, \quad z_2 = (1 \ 0) \begin{pmatrix} -1 \\ 1.75 \end{pmatrix} = -1, \quad z_3 = (1 \ 0) \begin{pmatrix} 1 \\ -1.25 \end{pmatrix} = 1, \quad z_4 = (1 \ 0) \begin{pmatrix} 1 \\ 0.75 \end{pmatrix} = 1$$

이제 [그림 6-19(c)]의 $\mathbf{u} = \left(\frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}}\right)^T$ 축으로 투영된 점을 구해 보자. $z_1 \sim z_4$ 의 분산은 1.0930이다.

$$z_1 = \left(\frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}}\right) \begin{pmatrix} -1 \\ -1.25 \end{pmatrix} = -1.591, \quad z_2 = \left(\frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}}\right) \begin{pmatrix} -1 \\ 1.75 \end{pmatrix} = 0.530,$$

$$z_3 = \left(\frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}}\right) \begin{pmatrix} 1 \\ -1.25 \end{pmatrix} = -0.177, \quad z_4 = \left(\frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}}\right) \begin{pmatrix} 1 \\ 0.75 \end{pmatrix} = 1.237$$

따라서 $\mathbf{u} = \left(\frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}}\right)^T$ 축이 $\mathbf{u} = (1 \ 0)^T$ 보다 우수하다고 할 수 있다. 그렇다면 $\mathbf{u} = \left(\frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}}\right)^T$ 보다 더 좋은 축이 있을까? 이제부터 최적해를 찾는 방법을 살펴보자.

6.6.1 주성분 분석

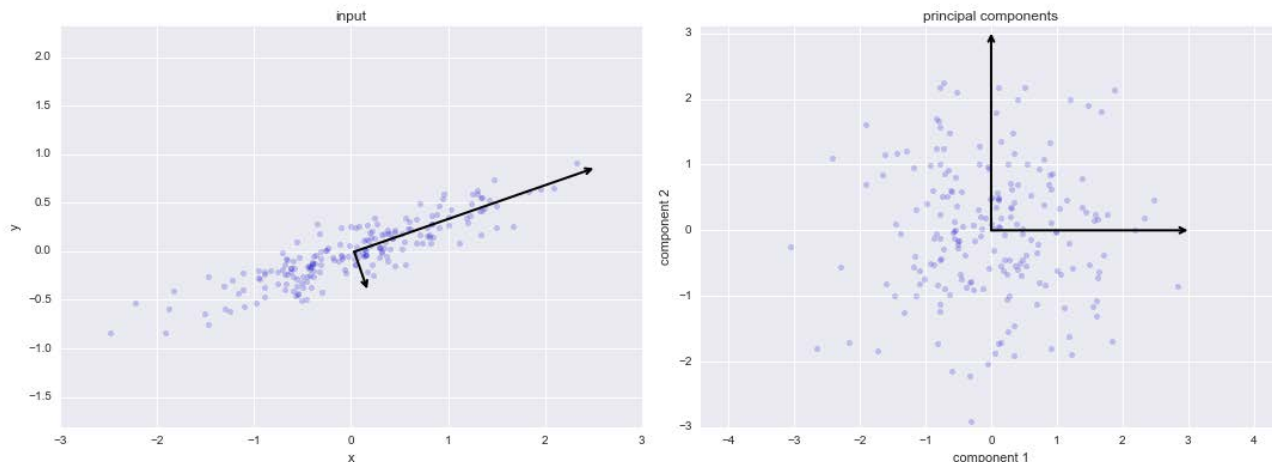
■ PCA의 최적화 문제

문제 6.1 $\mathbb{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n\}$ 의 분산을 최대화하는 q 개의 축, 즉 $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_q$ 를 찾아라. 이 단위 벡터는 식 (6.20)에 따라 변환 행렬 \mathbf{W} 를 구성한다.

- $q = 1$ 로 국한하고 분산을 쓰면,

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (z_i - \bar{z})^2 = \frac{1}{n} \sum_{i=1}^n z_i^2 = \frac{1}{n} \sum_{i=1}^n (\mathbf{u}^T \mathbf{x}_i)^2 = \mathbf{u}^T \Sigma \mathbf{u} \quad (6.21)$$

- [문제 6.1]을 바꾸어 쓰면, **문제 6.2** 식 (6.21)의 분산 σ^2 을 최대로 하는 \mathbf{u} 를 찾아라.



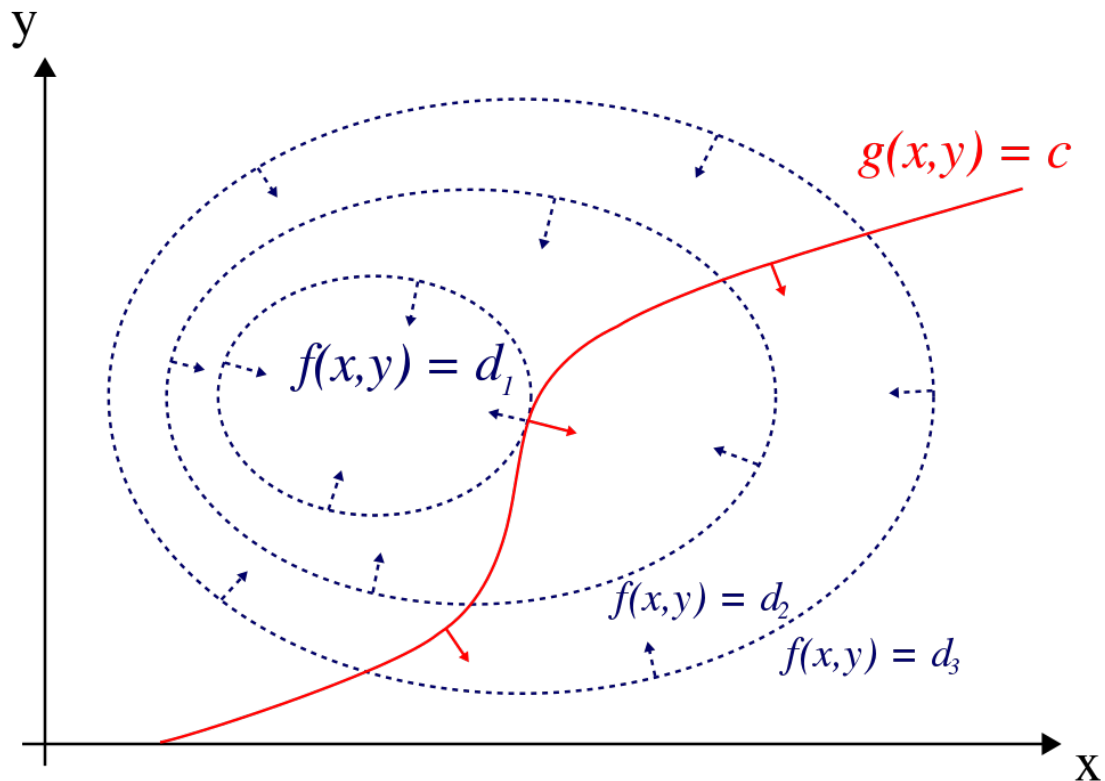
6.6.1 주성분 분석

■ \mathbf{u} 는 단위벡터($\mathbf{u}^T \mathbf{u} = 1$)라는 조건을 가지는 최대화 문제

- 조건이 있는 최적화 문제는 라그랑주 승수(Lagrangian multiplier)를 이용

Original problem: maximize $f(\cdot)$ subject to $g(\cdot) = c$

Reformulated with Lagrangian multiplier: maximize $f(\cdot) + \lambda(g(\cdot) - c)$



6.6.1 주성분 분석

■ PCA의 최적화 문제

- \mathbf{u} 가 단위 벡터라는 사실을 적용하여 문제를 다시 쓰면,

문제 6.3 $L(\mathbf{u}) = \mathbf{u}^T \Sigma \mathbf{u} + \lambda(1 - \mathbf{u}^T \mathbf{u})$ 를 최대로 하는 \mathbf{u} 를 찾아라.

- $L(\mathbf{u})$ 를 \mathbf{u} 로 미분하면, $\frac{\partial L(\mathbf{u})}{\partial \mathbf{u}} = 2\Sigma \mathbf{u} - 2\lambda \mathbf{u}$
- $\frac{\partial L}{\partial \mathbf{u}} = 0$ 을 풀면,

$$\Sigma \mathbf{u} = \lambda \mathbf{u} \quad (6.22)$$

■ 주성분 분석의 학습 알고리즘

1. 훈련집합으로 공분산 행렬 Σ 를 계산한다.
2. 식 (6.22)를 풀어 d 개의 고윳값과 고유 벡터를 구한다.
3. 고윳값이 큰 순서대로 $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \dots, \mathbf{u}_d$ 를 나열한다. (이들을 주성분이라 부름)
4. q 개의 주성분 $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_q$ 를 선택하여 식 (6.20)에 있는 행렬 \mathbf{W} 에 채운다.

6.6.1 주성분 분석

예제 6-3

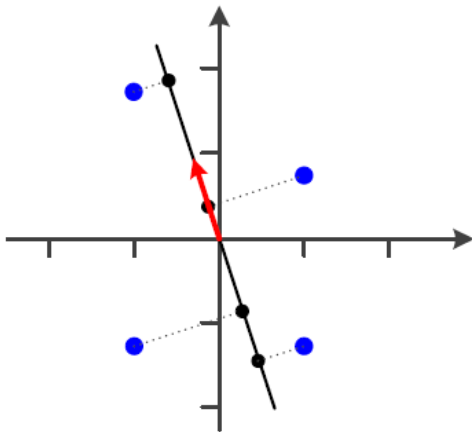
PCA 수행

식 (6.22)를 풀어 [그림 6-18]에 있는 데이터의 최적해를 구해 보자. 먼저 공분산 행렬 Σ 와 Σ 의 고윳값과 고유 벡터를 구하면 다음과 같다. 공분산을 구하는 방법은 2장의 식 (2.39)를 참조하라.

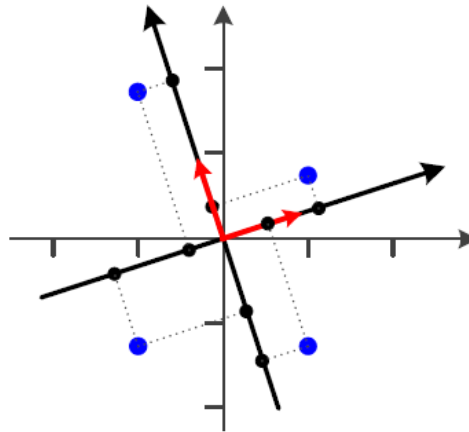
$$\Sigma = \begin{pmatrix} 1.000 & -0.250 \\ -0.250 & 1.688 \end{pmatrix}$$

$$\lambda_1 = 1.7688, \mathbf{u}_1 = \begin{pmatrix} -0.3092 \\ 0.9510 \end{pmatrix}, \lambda_2 = 0.9187, \mathbf{u}_2 = \begin{pmatrix} -0.9510 \\ -0.3092 \end{pmatrix}$$

고유 벡터 2개 중 고윳값이 큰 \mathbf{u}_1 을 선택하고, \mathbf{u}_1 에 샘플 4개를 투영하면 [그림 6-20(a)]가 된다. 변환된 점의 분산은 1.7688로 [그림 6-19]에 있는 축보다 훨씬 크다는 사실을 확인할 수 있다. \mathbf{u}_1 은 PCA 알고리즘으로 찾은 최적으로서 더 좋은 축은 없다.



(a) 축 1개 사용



(b) 축 2개 사용

그림 6-20 PCA가 찾은 최적 변환

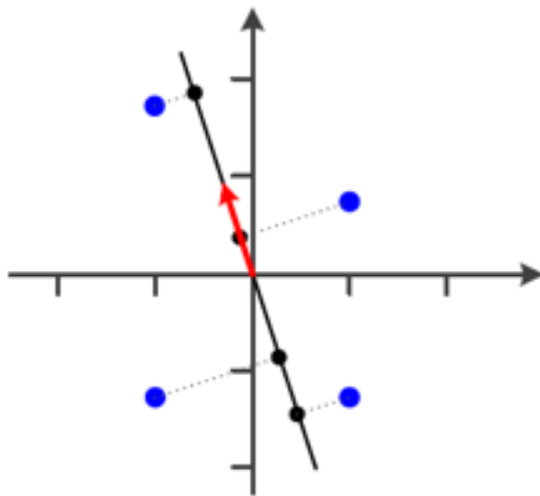
6.6.1 주성분 분석

■ 디코딩 과정

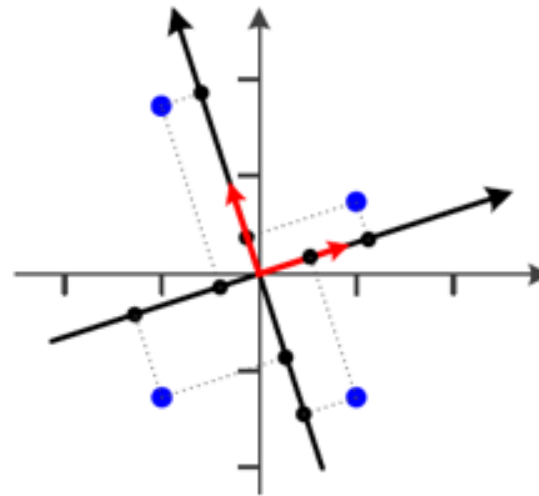
- 역변환은 $\mathbf{x} = (\mathbf{W}^T)^{-1} \mathbf{z}$ 인데, \mathbf{W} 가 정규직교 행렬($\mathbf{W}^T = \mathbf{W}^{-1}$ 성립)이므로 식 (6.23)이 됨

$$\tilde{\mathbf{x}} = \mathbf{W} \mathbf{z} \quad (6.23)$$

- $q = d$ 로 설정하면 \mathbf{W} 가 $d * d$ 이고 $\tilde{\mathbf{x}}$ 는 원래 샘플 \mathbf{x} 와 같게 됨([그림 6-20(b)]의 예시)
 - 원래 공간을 단지 일정한 양만큼 회전하는 것에 불과



(a) 축 1개 사용



(b) 축 2개 사용

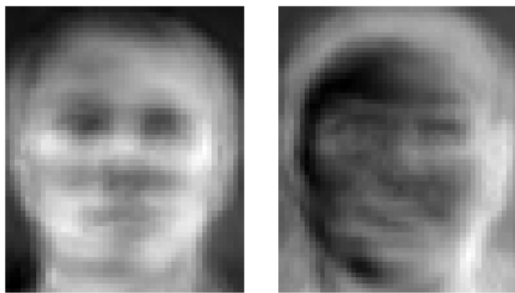
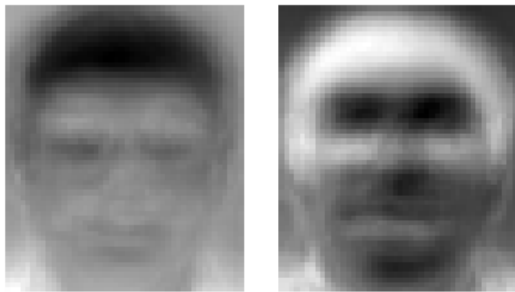
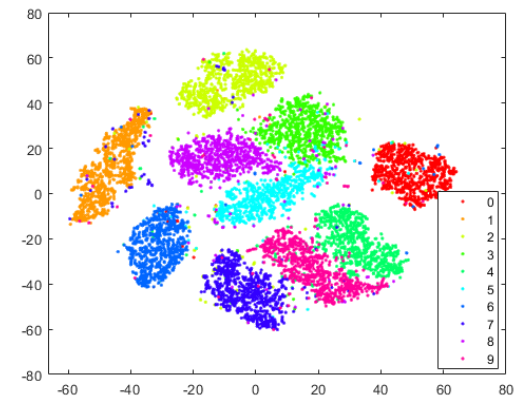
그림 6-20 PCA가 찾은 최적 변환

6.6.1 주성분 분석

■ 실제로는 $q < d$ 로 설정하여 차원 축소를 피함

■ 많은 응용이 있음

- 데이터 압축
- $q = 2$ 또는 $q = 3$ 으로 설정하여 2차원 또는 3차원으로 축소하여 데이터 가시화
- 고유얼굴 기법: 256×256 얼굴 영상($d = 65536$)을 $q = 7$ 차원으로 변환하여 얼굴 인식(정면 얼굴에 대해 96% 정확률) → 상위 몇 개의 고유 벡터가 대부분 정보를 가짐



Eigen Faces



6.6.2 독립 성분 분석

■ 블라인드 원음 분리 문제

- 실제 세계에서는 여러 신호가 섞여 나타남([그림 6-21]은 음악과 대화가 섞이는 예)

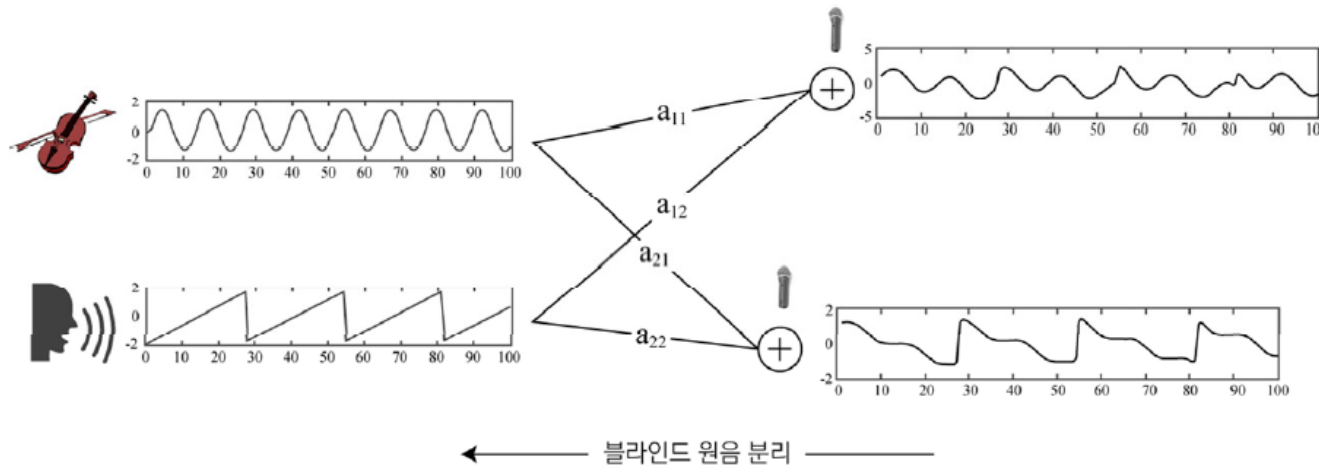


그림 6-21 블라인드 원음 분리 문제

- 마이크로 측정한 혼합 신호로부터 원음(음악과 목소리)을 복원할 수 있나? → 블라인드 원음 분리 문제라 부르며 독립 성분 분석 기법으로 해결 가능
- 아주 많은 예, 뇌파와 다른 장기 신호가 섞인 EEG, 장면과 잡음이 섞인 영상, ...

6.6.2 독립 성분 분석

■ 문제 정의

■ 표기

- 원래 신호를 $z_1(t)$ 와 $z_2(t)$, 측정된 혼합 신호를 $x_1(t)$ 와 $x_2(t)$ 로 표기
- t 순간에 획득된 $\mathbf{x}_t = (x_1(t), x_2(t))^T$ 를 훈련 샘플로 취함. 따라서 훈련집합은 $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$
- 블라인드 원음 분리 문제는 $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ 로부터 $\mathbb{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n\}$ 를 찾는 문제

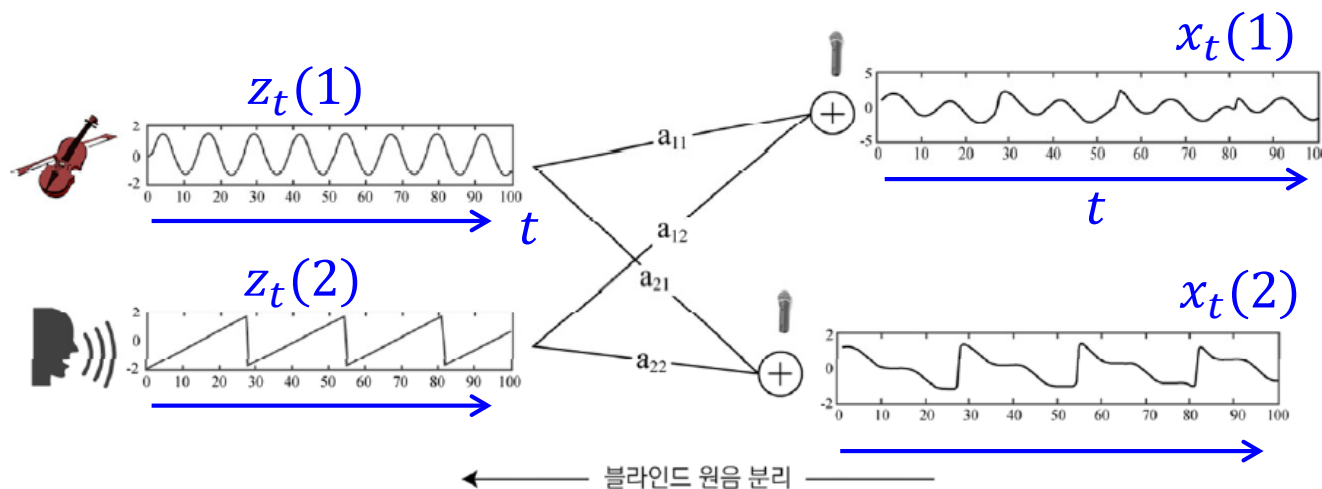


그림 6-21 블라인드 원음 분리 문제

6.6.2 독립 성분 분석

■ 문제 공식화

- 혼합 신호 \mathbf{x} 를 원래 신호 \mathbf{z} 의 선형 결합으로 표현 가능($z_1(t)$ 와 $z_2(t)$ 가 독립이라는 가정)

$$\begin{cases} x_1 = a_{11}z_1 + a_{12}z_2 \\ x_2 = a_{21}z_1 + a_{22}z_2 \end{cases} \quad (6.24)$$

- 행렬 표기로 쓰면,

$$\mathbf{x} = \mathbf{A}\mathbf{z} \quad (6.25)$$

- 블라인드 원음 분리 문제란 \mathbf{A} 를 구하는 것. \mathbf{A} 를 알면, 식 (6.26)으로 원음 복원

$$\tilde{\mathbf{z}} = \mathbf{W}\mathbf{x}, \quad \text{이때 } \mathbf{W} = \mathbf{A}^{-1} \quad (6.26)$$

■ 식 (6.25)는 과소 조건 문제 (ill-posed problem)

- 정수 하나를 주고 어떤 두 수의 곱인지 알아내라는 문제와 비슷함 (예를 들어, 32는 1×32 , 2×16 , 4×8 등 여러 답이 가능) \leftarrow 추가 조건을 주면 유일 해가 가능
- 문제도 과소 적합
- 추가 조건을 이용하여 식 (6.25)의 해를 찾음 \rightarrow 독립성 가정과 비가우시안 가정

6.6.2 독립 성분 분석

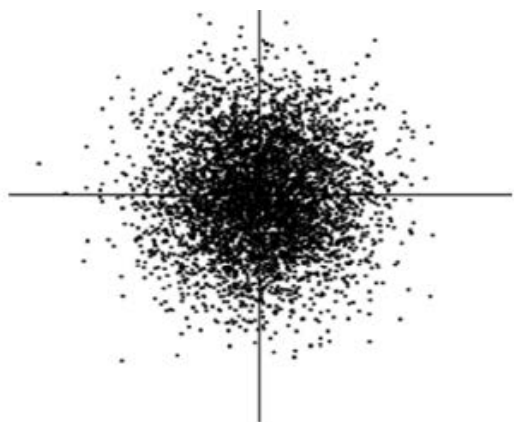
■ 독립성 가정

- 원래 신호가 서로 독립이라는 가정(예, 음악과 대화는 서로 무관하게 발생함)

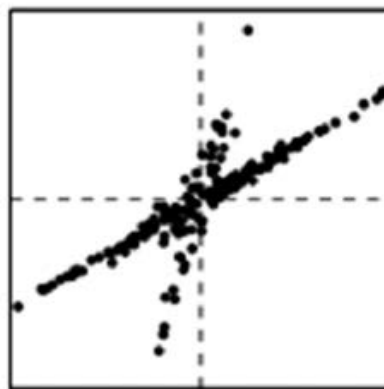
$$P(\mathbf{z}) = P(z_1, z_2, \dots, z_d) = \prod_{j=1}^d P(z_j) \quad (6.27)$$

■ 비가우시안 가정

- 원래 신호가 가우시안이라면 혼합 신호도 ([그림 6-22(a)]처럼) 가우시안이 되므로 분리할 실마리 없음. 비가우시안이면 ([그림 6-22(b)]처럼) 실마리가 있음



(a) 확률변수가 가우시안일 때



(b) 확률변수가 비가우시안일 때

그림 6-22 서로 독립인 두 확률변수의 결합 분포

6.6.2 독립 성분 분석

ICA의 문제 풀이

- 원래 신호의 비가우시안인 정도를 최대화하는 가중치를 구하는 전략 사용
 - 원래 신호를 식으로 쓰면,

$$\left. \begin{array}{l} z_j = w_{j1}x_1 + w_{j2}x_2 \\ \text{행렬 형태로 쓰면 } z_j = \mathbf{w}_j \mathbf{x} \end{array} \right\} \quad (6.28)$$

- 비가우시안을 최대화하는 가중치를 구하는 식을 쓰면,

$$\hat{\mathbf{w}}_j = \underset{\mathbf{w}_j}{\operatorname{argmax}} \check{G}(z_j) \quad (6.29)$$

- \check{G} 는 비가우시안 정도를 측정하는 함수
- 주로 식 (6.31)의 첨도(Kurtosis)를 사용

$$\text{kurtosis}(z_j) = \frac{1}{n} \sum_{i=1}^n z_{ji}^4 - 3 \left(\frac{1}{n} \sum_{i=1}^n z_{ji}^2 \right)^2 \quad (6.31)$$

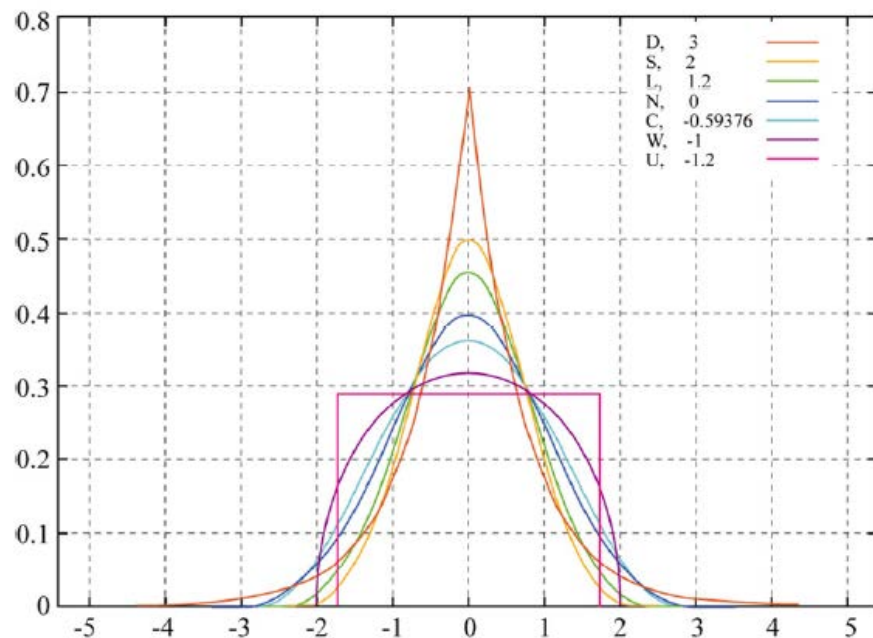


그림 6-23 여러 가지 분포의 첨도 측정

6.6.2 독립 성분 분석

■ ICA 학습

1. 전처리 수행

- 훈련집합 \mathbf{x} 의 평균이 $\mathbf{0}$ 이 되도록 이동(식 (6.19) 적용)
- 식 (6.30)의 화이트닝 (PCA를 통해 uncorrelated 된 정규 분포 생성)변환 적용

$$\mathbf{x}'_i = \left(\mathbf{D}^{-\frac{1}{2}} \mathbf{V}^T \right) \mathbf{x}_i, i = 1, 2, \dots, n \quad (6.30)$$

2. 식 (6.29)를 풀어 최적 가중치 구함

■ PCA와 ICA 비교

- ICA는 비가우시안과 독립성 가정, PCA는 가우시안과 비상관을 가정
- ICA는 4차 모멘트까지 사용, PCA는 2차 모멘트까지 사용
- ICA로 찾은 축은 수직 아님, PCA로 찾은 축은 서로 수직
- ICA는 주로 블라인드 원음 분리 문제를 푸는데, PCA는 차원 축소 문제를 푼다

6.6.3 희소 코딩(sparse coding)

- 기저함수(basis function) 또는 기저 벡터(basis vector)의 선형 결합으로 신호를 표현

- 푸리에 변환([그림 6-24(a)) 또는 웨이블릿 변환 등

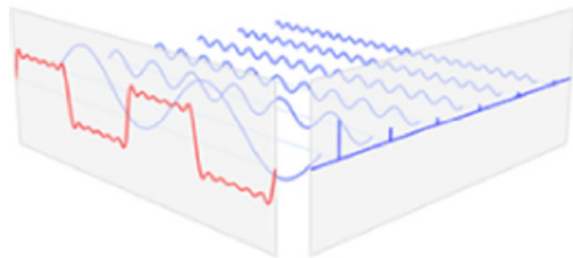
- 희소 코딩

- 사전 \mathbf{D} 를 구성하는 기저 벡터(단어) $\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_m$ 의 선형 결합으로 신호(영상) \mathbf{x} 를 표현

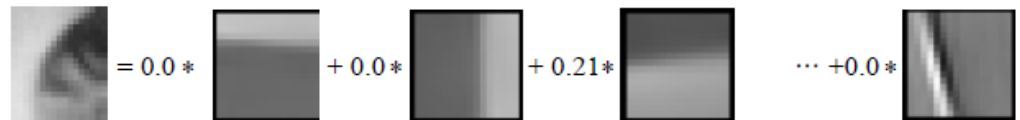
$$\mathbf{x} = \mathbf{D}\mathbf{a}$$

이때 $\mathbf{D} = (\mathbf{d}_1 \ \mathbf{d}_2 \ \dots \ \mathbf{d}_m)$

(6.32)



(a) 푸리에 변환

The equation shows a grayscale image \mathbf{x} as a linear combination of basis images $\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_m$. The coefficients are $0.0, 0.0, 0.21, \dots, 0.0$.

사전 $\mathbf{D} = \left\{ \begin{array}{c} \text{[basis image 1]} \quad \text{[basis image 2]} \quad \text{[basis image 3]} \quad \dots \quad \text{[basis image m]} \end{array} \right\}$

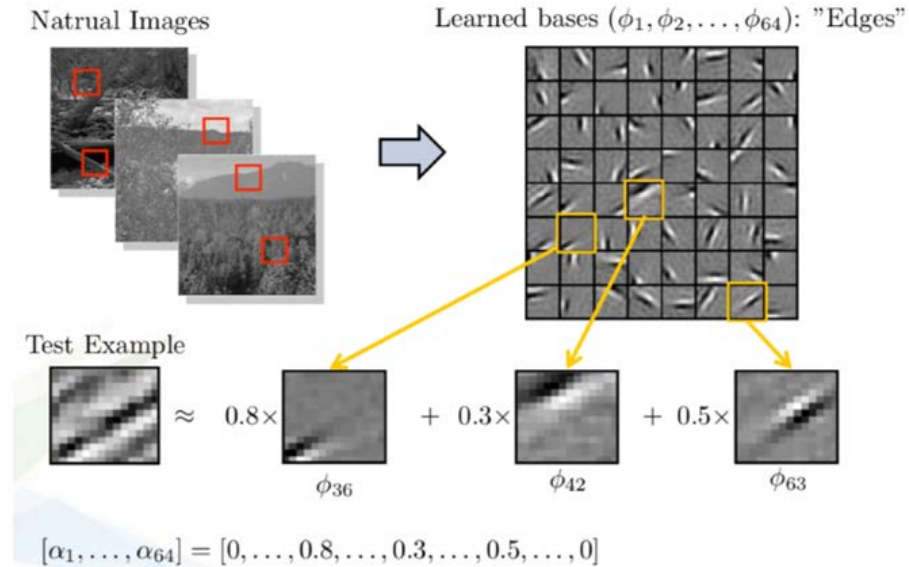
희소 코드 $\mathbf{a} = (0.0, 0.0, 0.21, \dots, 0.0)$

(b) 희소 코딩

그림 6-24 신호를 기저함수 또는 기저 벡터의 선형 결합으로 근사 표현

6.6.3 희소 코딩(sparse coding)

■ 희소 코딩의 예



■ 희소코딩 복원 방법

$$\begin{array}{c} \left[\begin{array}{c} \text{Input signal} \\ y_i \in R^{d \times 1} \end{array} \right] = \left[\begin{array}{c} \text{Over-complete dictionary} \\ D \in R^{d \times m} \end{array} \right] \left[\begin{array}{c} \text{Unknown} \\ \text{Sparse coefficient} \\ x_i \in R^{m \times 1} \end{array} \right] \end{array}$$

6.6.3 희소 코딩

■ 희소 코딩이 다른 변환 기법과 다른 점

- 비지도 학습이 사전(즉 기저 벡터)를 자동으로 알아냄(푸리에 변환은 삼각함수를 사용함)
→ 희소 코딩은 데이터에 맞는 기저 벡터를 학습해서 사용하는 셈
- 사전의 크기를 과잉 완벽(over complete)하게 책정($m > d$)
- 희소 코드 \mathbf{a} 를 구성하는 요소 대부분이 0 값을 가짐

■ 희소 코딩 구현

- 최적의 사전과 최적의 희소 코드를 알아내야 함
- ϕ 는 희소 코드의 희소성을 강제하는 규제항 (보통 L_1 놈 사용)

$$\hat{\mathbf{D}}, \hat{\mathbf{A}} = \operatorname{argmin}_{\mathbf{D}, \mathbf{A}} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{D}\mathbf{a}_i\|_2^2 + \lambda \phi(\mathbf{a}_i) \quad (6.33)$$

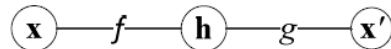
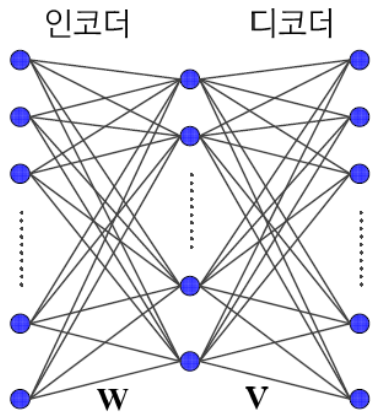
6.7 오토인코더

- 6.7.1 규제 오토인코더
- 6.7.2 적층 오토인코더

6.7 오토인코더

■ 오토인코더

- 특징 벡터 \mathbf{x} 를 입력 받아 동일한 또는 유사한 벡터 \mathbf{x}' 를 출력하는 신경망

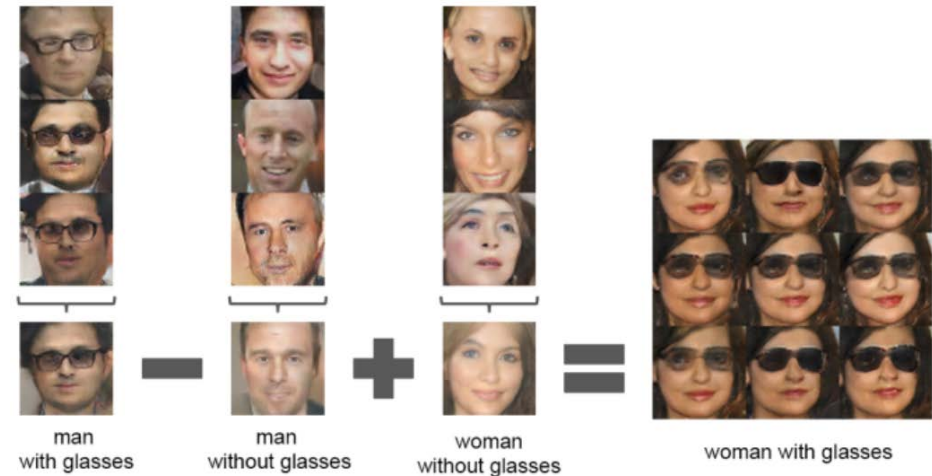


$\mathbf{x} \xrightarrow{f} \mathbf{h} \xrightarrow{g} \mathbf{x}'$

(a) 오토인코더의 구조와 동작

(b) 축약형

그림 6-25 오토인코더



Vector arithmetic for visual concepts

\mathbf{h} 조작의 예제

- 단순 복사하는 단위 행렬은 무의미

$$\mathbf{x} = g(f(\mathbf{x})) = \mathbf{VW}\mathbf{x} = \mathbf{I}\mathbf{x}$$

- 여러 가지 규제 기법 적용하여 유용한 신경망으로 활용

6.7 오토인코더

■ 병목 구조 오토인코더의 동작 원리

- $m < d$ 인 구조(예, 256*256 영상을 입력 받아 256*256 영상을 출력하는 경우 $d = 65536$ 인데 $m = 1024$ 로 설정)
- 은닉층의 \mathbf{h} 는 훨씬 적은 메모리로 데이터 표현. 필요하면 디코더로 원래 데이터 복원
- \mathbf{h} 는 \mathbf{x} 의 핵심 정보를 표현 \rightarrow 특징 추출, 영상 압축 등의 응용

■ 여러 형태의 오토인코더

- 은닉 노드의 개수에 따라 $m < d$, $m = d$, $m > d$ 구조
- 활성화함수에 따라 선형(식 (6.34))과 비선형(식 (6.35))

$$\left. \begin{aligned} \mathbf{h} &= f(\mathbf{x}) = \mathbf{W}\mathbf{x} \\ \mathbf{x} &= g(\mathbf{h}) = \mathbf{V}\mathbf{h} \end{aligned} \right\} \quad (6.34)$$

$$\left. \begin{aligned} \mathbf{h} &= f(\mathbf{x}) = \tau_{\text{encode}}(\mathbf{W}\mathbf{x}) \\ \mathbf{x} &= g(\mathbf{h}) = \tau_{\text{decode}}(\mathbf{V}\mathbf{h}) \end{aligned} \right\} \quad (6.35)$$

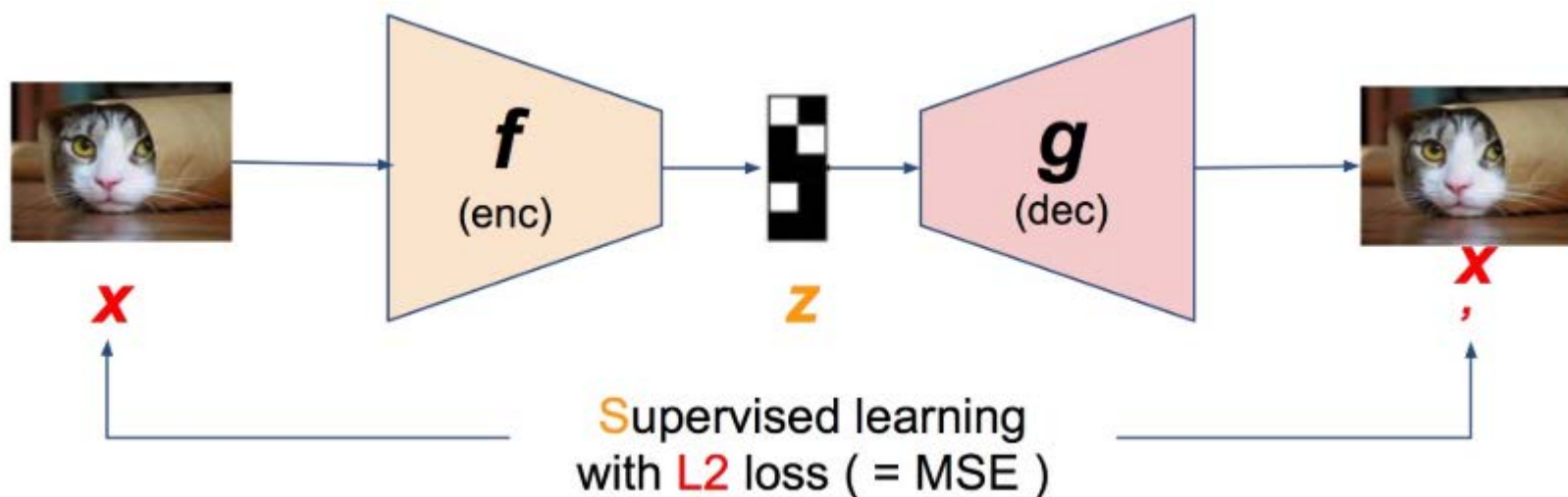
6.7 오토인코더

■ 오토인코더의 학습

- 주어진 데이터는 훈련집합 $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, 알아내야 하는 매개변수는 f 와 g 라는 매핑 함수, 즉 가중치집합 $\Theta = \{\mathbf{W}, \mathbf{V}\}$
- 오토인코더 학습을 최적화 문제로 쓰면,

$$\hat{\Theta} = \operatorname{argmin}_{\Theta} \sum_{i=1}^n L(\mathbf{x}_i, g(f(\mathbf{x}_i))) \quad (6.36)$$

$$L(\mathbf{x}_i, g(f(\mathbf{x}_i))) = \|\mathbf{x}_i - g(f(\mathbf{x}_i))\|_2^2 \quad (6.37)$$



6.7.1 규제 오토인코더

■ 여러 규제 기법을 적용

- $m > d$ 인 상황에서도 단순 복사를 피할 수 있음 ← 충분히 큰 모델을 사용하되 적절한 규제 기법을 적용하는 현대 기계 학습 추세를 오토인코더로 따르는 셈

■ SAE(sparse autoencoder)

- 은닉 벡터 \mathbf{h}_i 가 희소하도록 강제화(0이 아닌 요소의 개수를 적게 유지)
- $\phi(\mathbf{h}_i)$ 는 벡터 \mathbf{h}_i 가 희소하도록 강제하는 규제항

$$\text{SAE: } \hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} \sum_{i=1}^n L(\mathbf{x}_i, g(f(\mathbf{x}_i))) + \lambda \phi(\mathbf{h}_i) \quad (6.38)$$

■ DAE(denoising autoencoder)

- 잡음을 추가한 다음 원본을 복원하도록 학습하는 원리
- 특징 벡터 \mathbf{x}_i 에 적절한 양의 잡음을 추가한 $\tilde{\mathbf{x}}_i$ 를 입력으로 사용

$$\text{DAE: } \hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} \sum_{i=1}^n L(\mathbf{x}_i, g(f(\tilde{\mathbf{x}}_i))) \quad (6.39)$$

6.7.1 규제 오토인코더

■ 다양한 응용

- 영상 복원, 영상 편집
- 시각화, 군집화

