차원 축소/오토인코더

- 차원 축소

SECTION 8-1 차원의 저주
SECTION 8-2 차원 축소를 위한 접근 방법
SECTION 8-3 PCA
SECTION 8-4 커널 PCA
SECTION 8-5 LLE
SECTION 8-6 다른 차원 축소 기법
SECTION 8-7 연습문제

차원 축소

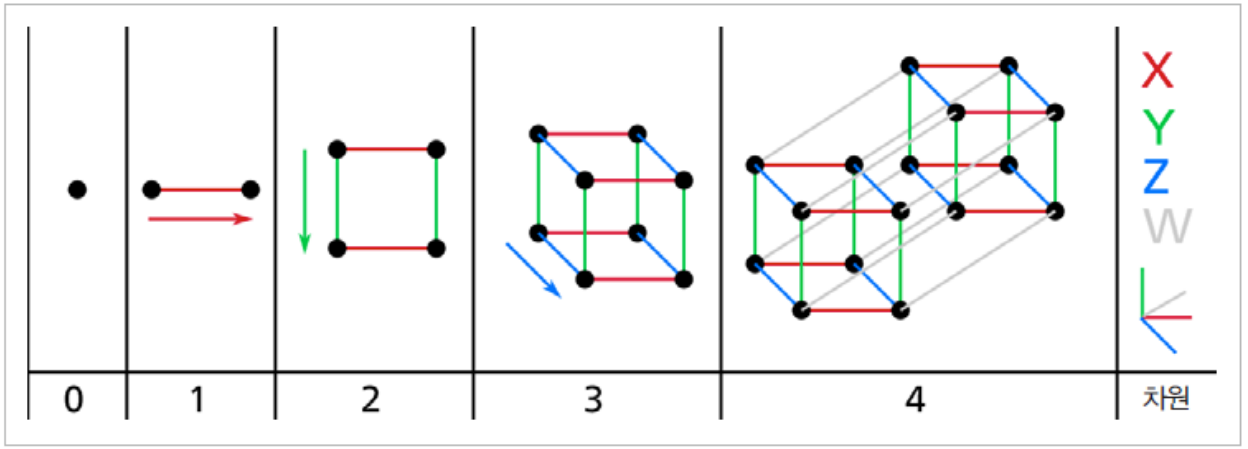
고차원 공간과 차원 축소 기법

SECTION 8-1 차원의 저주

- 차원의 저주
  훈련 샘플 각각이 수천 심지어 수백만 개의 특성을 가지고 있어서, 많은 특성은 훈련을 느리게 할 뿐만 아니라, 앞좋은 솔루션을 찾기 어렵게 만드는 문제.

- 우리는 3차원 세계에서 살고 있어서 고차원 공간은 직관적 상상 불가
  - 3차원 큐브에서 임의의 두 점을 선택하면 평균 거리는 대략 0.66. 만약 1,000,000차원의 초입방체에서 두 점을 무작위로 선택할 경우 평균 거리는? 약 428.25(대략 $\sqrt{(1000000 / 6)}$
  - 훈련 세트의 차원이 클수록 과대적합 위험이 커짐

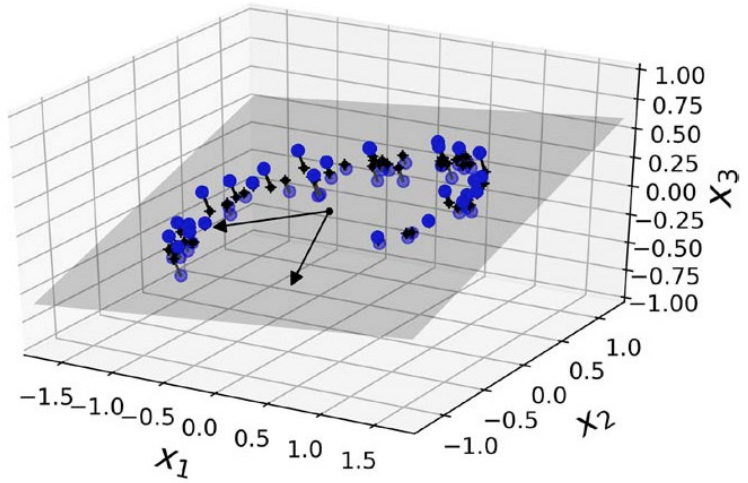
  - 어떻게 해결할까?
    - 이론적으로 차원의 저주를 해결하는 해결책 하나는 훈련 샘플의 밀도가 충분히 높아질 때까지 훈련 세트의 크기를 키우는 것. 불행하게도 실제로는 일정 밀도에 도달하기 위해 필요한 훈련 샘플 수는 차원 수가 커짐에 따라 기하급수적으로 늘어남



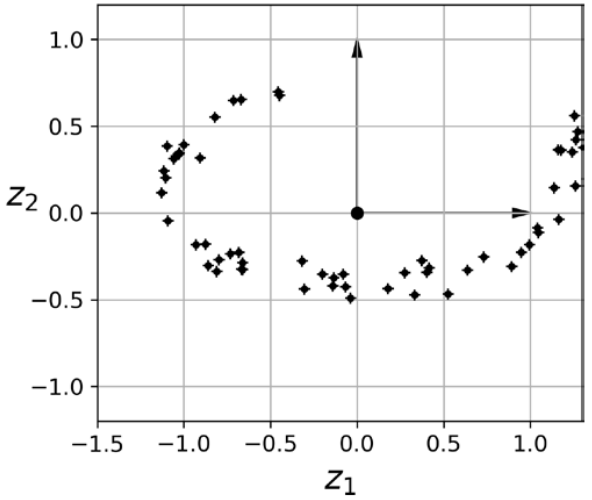
▲ 그림 8-1 점, 선, 정사각형, 정육면체, 테서랙트(0차원에서 4차원까지의 초입방체

SECTION 8-2 차원 축소를 위한 접근 방법(1)

- 차원을 감소시키는 두 가지 주요한 접근법인 투영과 매니폴드 학습
  - 투영
    - 대부분의 실전 문제는 훈련 샘플이 모든 차원에 걸쳐 균일하게 퍼져 있지 않음
    - 많은 특성은 거의 변화가 없는 반면, 다른 특성들은 서로 강하게 연관되어 결과적으로 모든 훈련 샘플이 고차원 공간 안의 저차원 부분 공간에(또는 가까이) 놓여 있음
    - 모든 훈련 샘플을 이 부분 공간에 수직으로(즉, 샘플과 평면 사이의 가장 짧은 직선을 따라) 투영하면 [그림 8-3]과 같은 2D 데이터셋을 얻음
    - 그러나 차원 축소에 있어서 투영이 언제나 최선의 방법은 아니며, 많은 경우 스위스 롤 데이터셋처럼 부분 공간이 뒤틀리거나 휘어 있기도 단순히 평면에 투영시키면 스위스 롤의 층이 서로 뭉개짐



▲ 그림 8-2 2차원에 가깝게 배치된 3차원 데이터셋



▲ 그림 8-3 투영하여 만들어진 새로운 2D 데이터셋

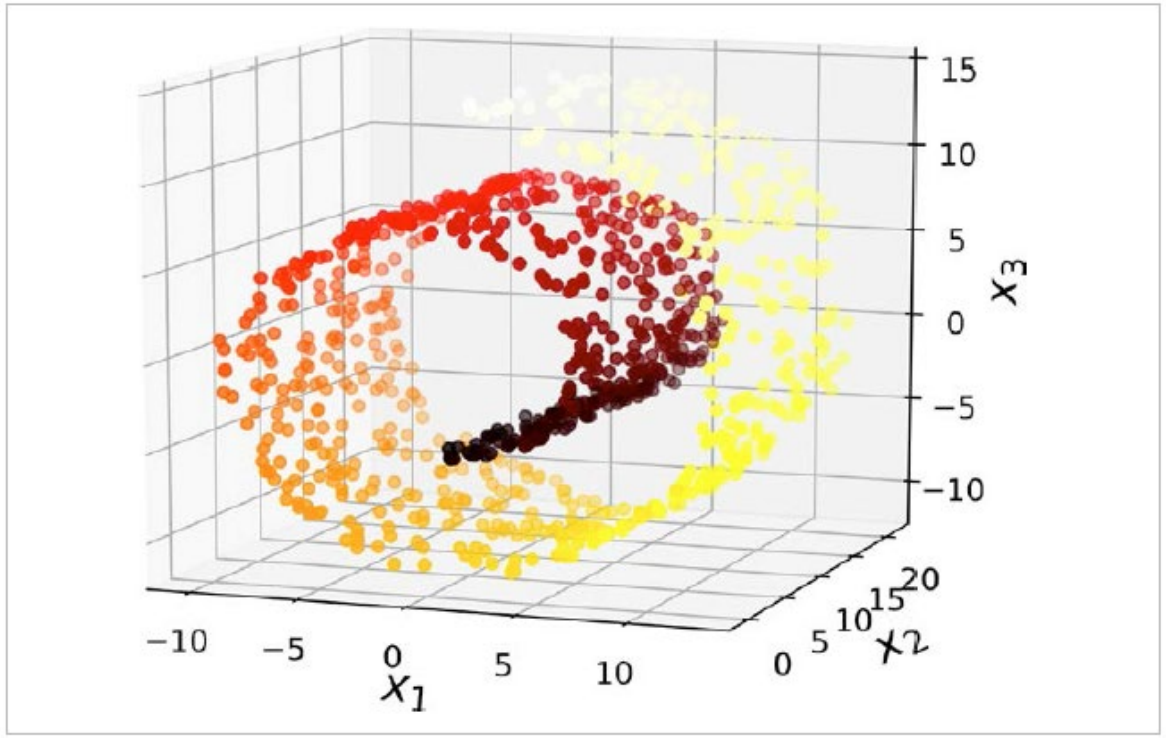


그림 8-4 스위스 롤 데이터셋

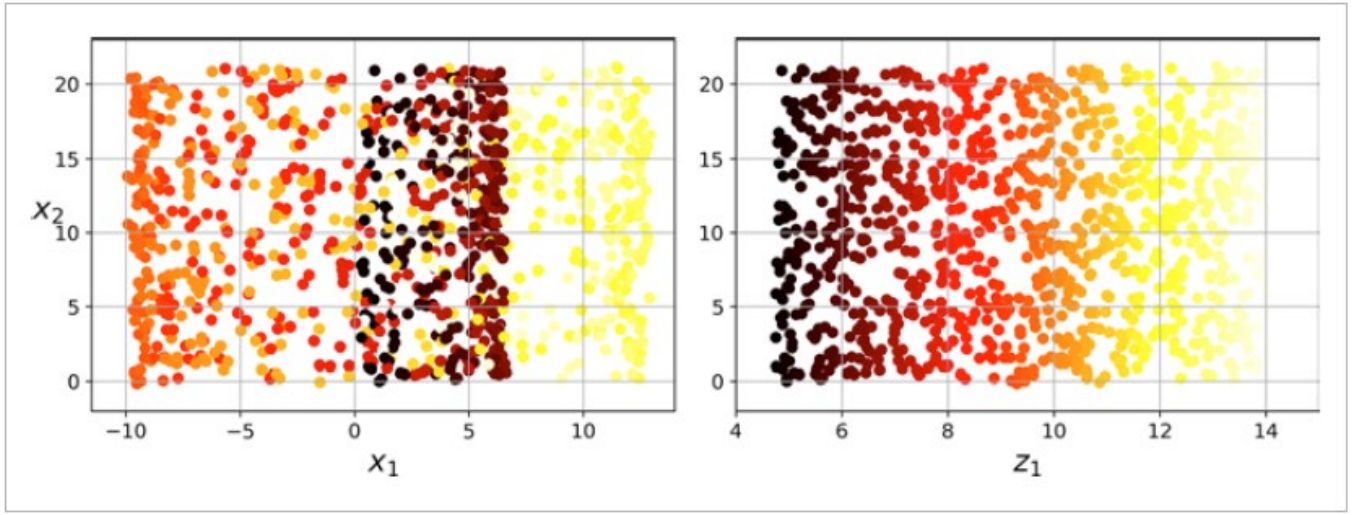
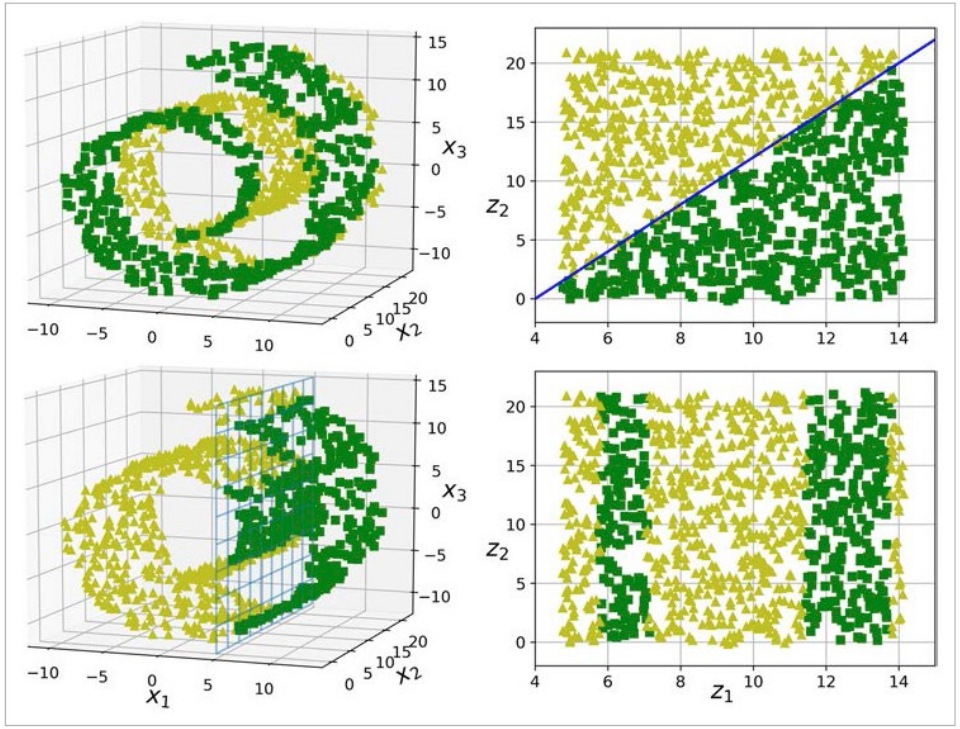


그림 8-5 평면에 그냥 투영시켜서 뭉개진 것(왼쪽)과 스위스 롤을 펼쳐놓은 것(오른쪽)

SECTION 8-2 차원 축소를 위한 접근 방법(2)

- 매니폴드 학습
  - 대부분 실제 고차원 데이터셋이 더 낮은 저차원 매니폴드에 가깝게 놓여 있다는 매니폴드 가정 또는 매니폴드 가설에 근거
  - 2D 매니폴드는 고차원 공간에서 휘어지거나 뒤틀린 2D 모양
  - *d*차원 매니폴드는 국부적으로 *d*차원 초평면으로 보일 수 있는 *n*차원 공간의 일부 ($d<n$)



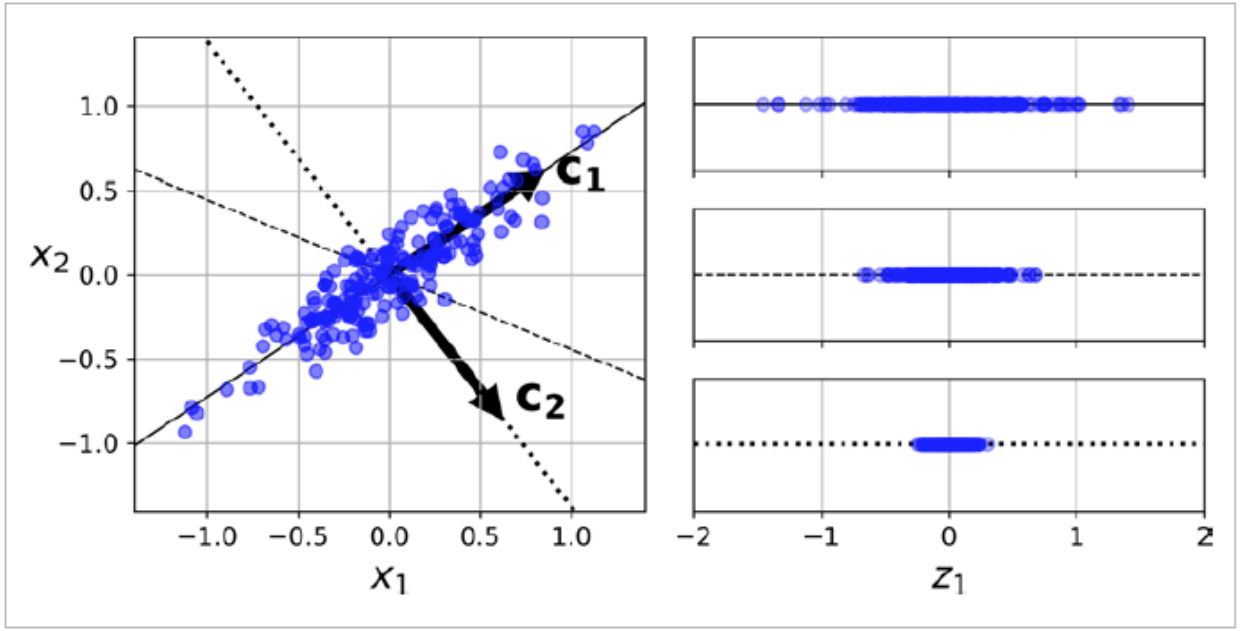
▲ 그림 8-6 저차원에서 항상 간단하지 않은 결정 경계

SECTION 8-3 PCA

- PCA: 주성분 분석(principal component analysis)
  - 먼저 데이터에 가장 가까운 초평면을 정의한 다음, 데이터를 이 평면에 투영

▪ 분산 보존

: 저차원의 초평면에 훈련 세트를 투영하기 전에 먼저 올바른 초평면을 선택해야 한다.
실선에 투영된 것은 분산을 최대로 보존하는 반면, 점선에 투영된 것은 분산을 매우 적게 유지한다.



▲ 그림 8-7 투영할 부분 공간 선택하기

SECTION 8-3 PCA

- PCA: 주성분 분석(principal component analysis)
  - 먼저 데이터에 가장 가까운 초평면을 정의한 다음, 데이터를 이 평면에 투영

▪ 주성분

식 8-1 주성분 행렬

$$\mathbf{V} = \begin{pmatrix} | & | & & | \\ \mathbf{c}_1 & \mathbf{c}_2 & \cdots & \mathbf{c}_n \\ | & | & & | \end{pmatrix}$$

▪ d차원으로 투영하기

식 8-2 훈련 세트를 d차원으로 투영하기

$$\mathbf{X}_{d\text{-proj}} = \mathbf{X}\mathbf{W}_d$$

SECTION 8-3 PCA

- PCA: 주성분 분석(principal component analysis)
  - 먼저 데이터에 가장 가까운 초평면을 정의한 다음, 데이터를 이 평면에 투영

  - 사이킷런 사용하기

```
from sklearn.decomposition import PCA

pca = PCA(n_components = 2)
X2D = pca.fit_transform(X)
```

  - 설명된 분산의 비율
  - 적절한 차원 수 선택하기

```
pca = PCA()
pca.fit(X_train)
cumsum = np.cumsum(pca.explained_variance_ratio_)
d = np.argmax(cumsum >= 0.95) + 1
```

SECTION 8-3 PCA

- PCA: 주성분 분석(principal component analysis)
  - 먼저 데이터에 가장 가까운 초평면을 정의한 다음, 데이터를 이 평면에 투영
  - 압축을 위한 PCA

식 8-3 원본의 차원 수로 되돌리는 PCA 역변환

$$\mathbf{X}_{\text{recovered}} = \mathbf{X}_{d\text{-proj}} \mathbf{W}_d^T$$

그림 8-9 분산의 95%가 유지된 MNIST 압축

  - 랜덤 PCA
  - 점진적 PCA

SECTION 8-4 커널 PCA

- 커널 PCA(kPCA)
  - 차원 축소를 위한 복잡한 비선형 투형을 수행.
  - 투영된 후에 샘플의 군집을 유지하거나 꼬인 매니폴드에 가까운 데이터셋을 펼칠 때도 유용함
- 커널 선택과 하이퍼파라미터 튜닝
  - kPCA는 비지도 학습이므로 좋은 커널과 하이퍼파라미터를 선택하기 위한 명확한 성능 측정 기준이 없음
  - 하지만 차원 축소는 종종 지도 학습(예를 들면 분류)의 전처리 단계로 활용되므로 그리드 탐색을 사용하여 주어진 문제에서 성능이 가장 좋은 커널과 하이퍼파라미터를 선택이 가능
  - 재구성 원상

SECTION 8-4 커널 PCA

◦ 커널 PCA(kPCA)

- 차원 축소를 위한 복잡한 비선형 투형을 수행.
- 투영된 후에 샘플의 군집을 유지하거나 꼬인 매니폴드에 가까운 데이터셋을 펼칠 때도 유용함

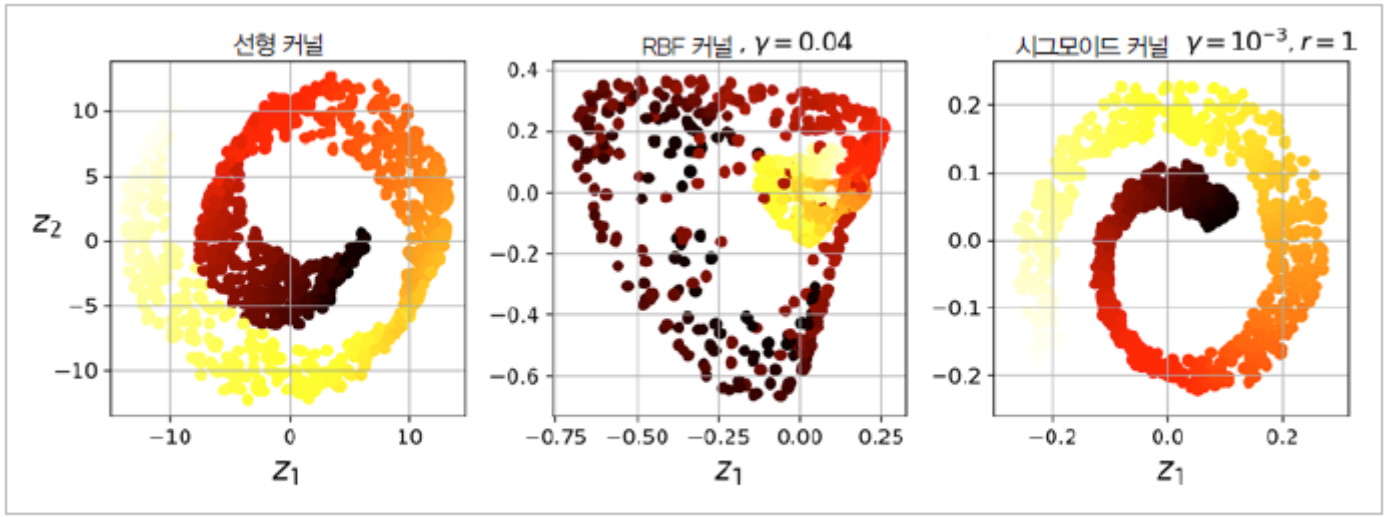


그림 8-10 여러 가지 커널의 kPCA를 사용해 2D로 축소시킨 스위스 롤

▪ 커널 선택과 하이퍼파라미터 튜닝

- kPCA는 비지도 학습이므로 좋은 커널과 하이퍼파라미터를 선택하기 위한 명확한 성능 측정 기준이 없음
- 하지만 차원 축소는 종종 지도 학습(예를 들면 분류)의 전처리 단계로 활용되므로 그리드 탐색을 사용하여 주어진 문제에서 성능이 가장 좋은 커널과 하이퍼파라미터를 선택이 가능
- 재구성 원상

SECTION 8-4 커널 PCA

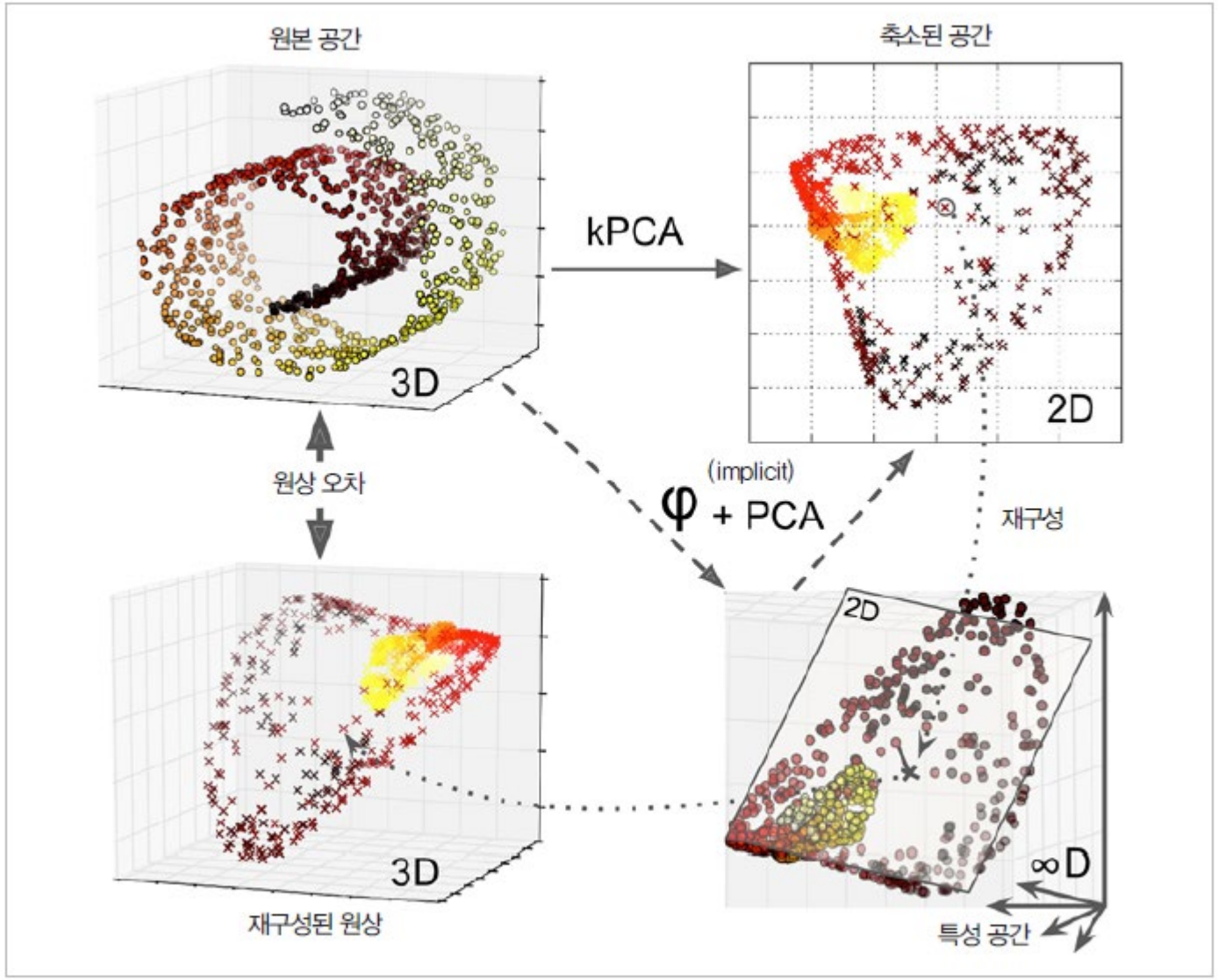


그림 8-11 커널 PCA와 재구성 원상 오차

SECTION 8-5 LLE

- 지역 선형 임베딩(locally linear embedding, LLE)
  - 강력한 비선형 차원 축소(NLDR) 기술, 이전 알고리즘처럼 투영에 의존하지 않는 매니폴드 학습
  - 먼저 각 훈련 샘플이 가장 가까운 이웃closest neighbor(c.n.)에 얼마나 선형적으로 연관되어 있는지 측정 → 국부적인 관계가 가장 잘 보존되는 훈련 세트의 저차원 표현 탐색
  - 특히 잡음이 너무 많지 않은 경우 꼬인 매니폴드를 펼치는 데 잘 작동함
  - 사이킷런의 LocallyLinearEmbedding을 사용해 스위스 롤을 펼쳐보기

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_{i=1}^{m} \left(\mathbf{x}^{(i)} - \sum_{j=1}^{m} w_{i,j} \mathbf{x}^{(j)} \right)^2$$

[조건] $\begin{cases} w_{i,j} = 0 & \mathbf{x}^{(j)}\text{가 } \mathbf{x}^{(i)}\text{의 최근접 이웃 } k\text{개 중 하나가 아닐 때} \\ \sum_{j=1}^{m} w_{i,j} = 1 & i = 1, 2, \cdots, m\text{일 때} \end{cases}$

▲식 8-4 LLE 단계 1: 선형적인 지역 관계 모델링

$$\mathbf{Z} = \underset{\mathbf{Z}}{\operatorname{argmin}} \sum_{i=1}^{m} \left(\mathbf{z}^{(i)} - \sum_{j=1}^{m} \hat{w}_{i,j} \mathbf{z}^{(j)} \right)^2$$

▲식 8-5 LLE 단계 2: 관계를 보존하는 차원 축소

SECTION 8-6 다른 차원 축소 기법

- 랜덤 투영(random projection)
  - 랜덤한 선형 투영을 사용해 데이터를 저차원 공간으로 투영
- 다차원 스케일링(MDS, multidimensional scaling)
  - 샘플 간의 거리를 보존하면서 차원을 축소
- Isomap
  - 각 샘플을 가장 가까운 이웃과 연결하는 식으로 그래프를 만들고, 샘플 간의 지오데식 거리(geodesic distance)를 유지하면서 차원을 축소
- t-SNE(t-distributed stochastic neighbor embedding)
  - 비슷한 샘플은 가까이, 비슷하지 않은 샘플은 멀리 떨어지도록 하면서 차원을 축소
- 선형 판별 분석(LDA, linear discriminant analysis)
  - 분류 알고리즘이지만 훈련 과정에서 클래스 사이를 가장 잘 구분하는 축을 학습. 이 축은 데이터가 투영되는 초평면을 정의하는 데 사용

SECTION 8-7 연습문제(1)

1. 데이터셋의 차원을 축소하는 주요 목적은 무엇인가요? 대표적인 단점은 무엇인가?
2. 차원의 저주란 무엇인가?
3. 데이터셋의 차원을 축소시키고 나서 이 작업을 원복할 수 있나? 할 수 있다면 어떻게 가능할까? 가능하지 않다면 왜일까?
4. 매우 비선형적인 데이터셋의 차원을 축소하는 데 PCA를 사용할 수 있을까?
5. 설명된 분산을 95%로 지정한 PCA를 1,000개의 차원을 가진 데이터셋에 적용한다고 가정하자. 결과 데이터셋의 차원은 얼마나 될까?
6. 기본 PCA, 점진적 PCA, 랜덤 PCA, 커널 PCA는 어느 경우에 사용될까?
7. 어떤 데이터셋에 적용한 차원 축소 알고리즘의 성능을 어떻게 평가할 수 있을까?
8. 두 개의 차원 축소 알고리즘을 연결할 수 있을까?

연습문제의 정답은 부록A에서 확인

SECTION 8-7 연습문제(2)

9. (3장에서 소개한) MNIST 데이터셋을 로드하고 훈련 세트와 테스트 세트로 분할(처음 60,000개는 훈련을 위한 샘플이고 나머지 10,000개는 테스트용).
   이 데이터셋에 랜덤 포레스트 분류기를 훈련시키고 얼마나 오래 걸리는지 시간을 잰 다음, 테스트 세트로 만들어진 모델을 평가.
   그런 다음 PCA를 사용해 설명된 분산이 95%가 되도록 차원을 축소하고, 이 축소된 데이터셋에 새로운 랜덤 포레스트 분류기를 훈련시키고 얼마나 오래 걸리는지 확인.
   훈련 속도가 더 빨라졌는가? 이제 테스트 세트에서 이 분류기를 평가해보자. 이전 분류기와 비교해서 어떤가?

10. t-SNE 알고리즘을 사용해 MNIST 데이터셋을 2차원으로 축소시키고 맷플롯립으로 그래프를 그리기. 이미지의 타깃 클래스마다 10가지 색상으로 나타낸 산점도를 그릴 수 있음. 또는 산점도의 각 포인트를 이에 상응하는 샘플의 클래스(0에서 9까지 숫자)로 바꾸거나 숫자 이미지 자체의 크기를 줄여서 그릴 수도 있음(모든 숫자를 다 그리면 그래프가 너무 복잡해지므로 무작위로 선택한 샘플만 그리거나, 인접한 곳에 다른 샘플이 그려져 있지 않은 경우에만 그림). 잘 분리된 숫자의 군집을 시각화할 수 있을 것임.
    PCA, LLE, MDS 같은 차원 축소 알고리즘을 적용해보고 시각화 결과를 비교해보기

연습문제의 정답은 부록A에서 확인

Contents

- CHAPTER 17: 오토인코더와 GAN을 사용한 표현 학습과 생성적 학습

오토인코더

오토인코더를 사용한 비지도 방식의 심층 표현 방법

SECTION 17-1 효율적인 데이터 표현

- 오토인코더
  - 어떤 지도 없이도(즉, 레이블되어 있지 않은 훈련 데이터를 사용해서) 잠재 표현 또는 코딩이라 부르는 입력 데이터의 밀집 표현을 학습할 수 있는 인공 신경망
  - 생성적 적대 신경망(GAN, generative adversarial networks)
  - 다음과 같은 숫자 시퀀스를 쉽게 기억할 수 있는 방법이 있을까?
    - 40, 27, 25, 36, 81, 57, 10, 73, 19, 68
    - 50, 48, 46, 44, 42, 40, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, 18, 16, 14
  - 오토인코더에 제약을 가해서 데이터에 있는 패턴을 찾아 활용
  - 재구성
  - 재구성 손실
  - 과소완전

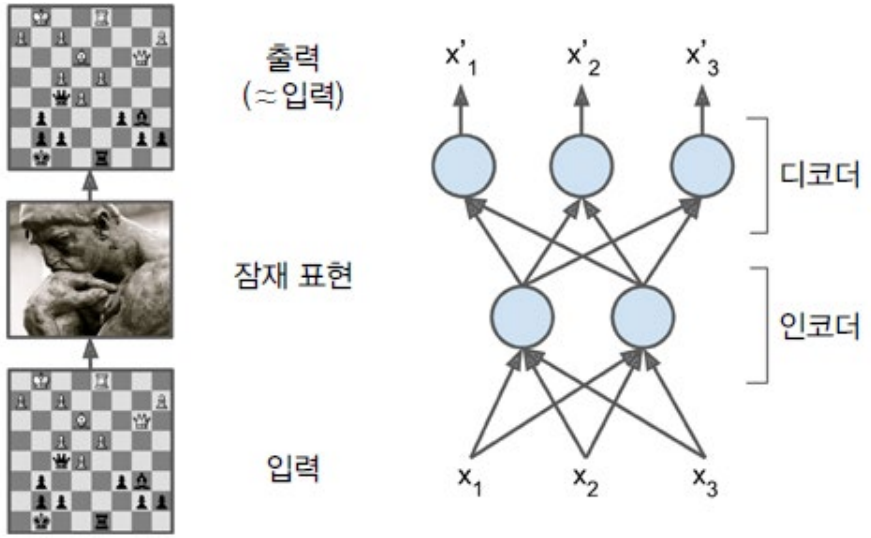
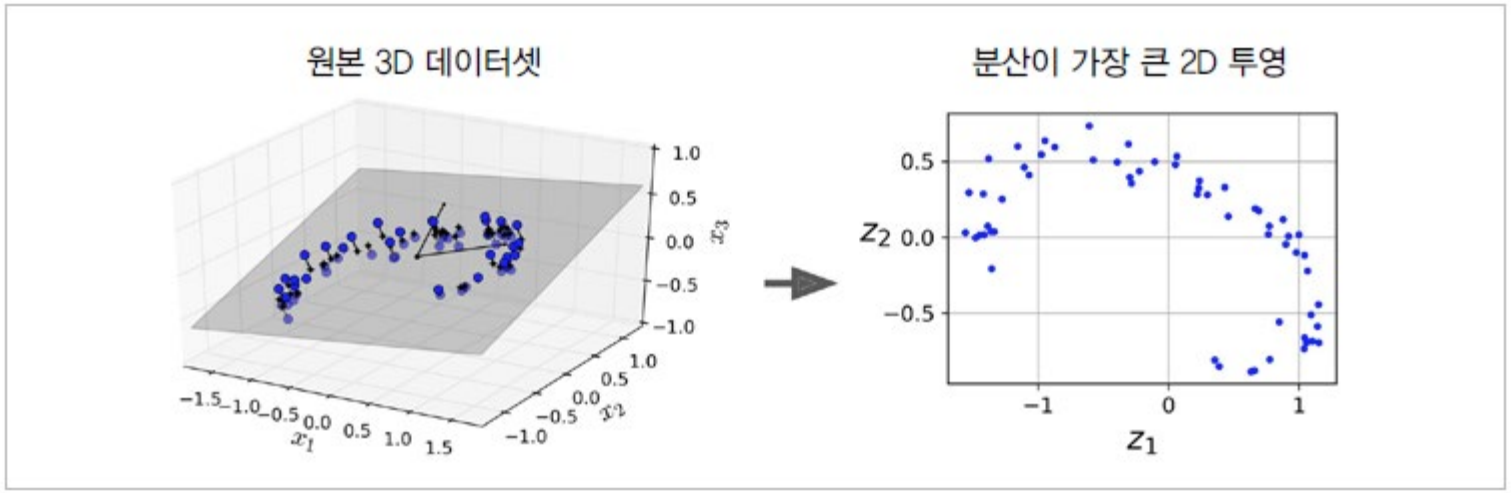


그림 17-1 체스 기억 실험(왼쪽)과 간단한 오토인코더(오른쪽) ▶

SECTION 17-2 과소완전 선형 오토인코더로 PCA 수행하기

- 오토인코더가 선형 활성화 함수만 사용하고 비용 함수가 평균 제곱 오차(MSE)라면, 이는 결국 주성분 분석(PCA, 8장 참조)을 수행하는 것으로 볼 수 있음
  - 오토인코더 수행 학습
    - 3D 데이터셋에 PCA를 적용해 2D에 투영하는 간단한 선형 오토인코더 생성
    - 가상으로 생성한 간단한 3D 데이터셋에 훈련
    - 동일한 데이터셋을 인코딩(즉 2D로 투영)

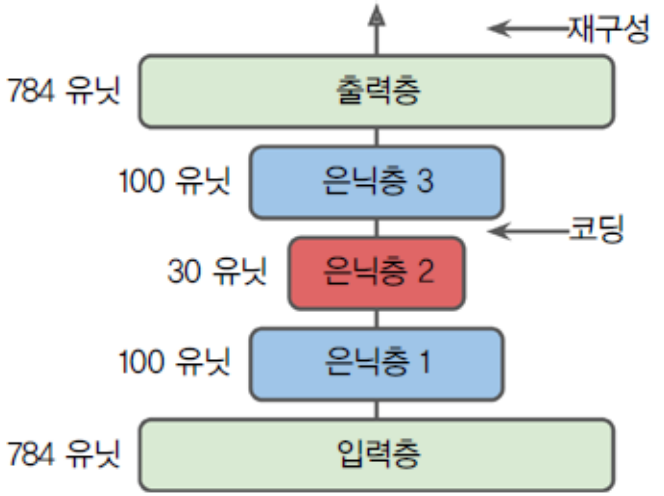


▲ 그림 17-2 과소완전 선형 오토인코더로 수행한 PCA

SECTION 17-3 적층 오토인코더

- 적층 오토인코더 (또는, 심층 오토인코더)
  - 여러 개의 은닉층을 가진 오토인코더
- 케라스를 사용하여 적층 오토인코더 구현하기
  - 일반적인 심층 MLP와 매우 비슷하게 적층 오토인코더를 구현
- 재구성 시각화
  - 입력과 출력을 비교하여 오토인코더가 적절히 훈련되었는지 확인

그림 17-4 원본 이미지(위)와 재구성(아래)



▲ 그림 17-3 적층 오토인코더

SECTION 17-3 적층 오토인코더

- 적층 오토인코더 (또는, 심층 오토인코더)
  - 패션 MNIST 데이터셋 시각화
    - 적절한 수준으로 차원을 축소한 후 다른 차원 축소 알고리즘을 사용해 시각화
  - 적층 오토인코더를 사용한 비지도 사전훈련
    - 기존의 네트워크에서 학습한 특성 감지 기능을 재사용
  - 가중치 묶기
  - 한 번에 오토인코더 한 개씩 훈련하기
    - 탐욕적 방식의 층별 훈련 (greedy layerwise training)

SECTION 17-3 적층 오토인코더

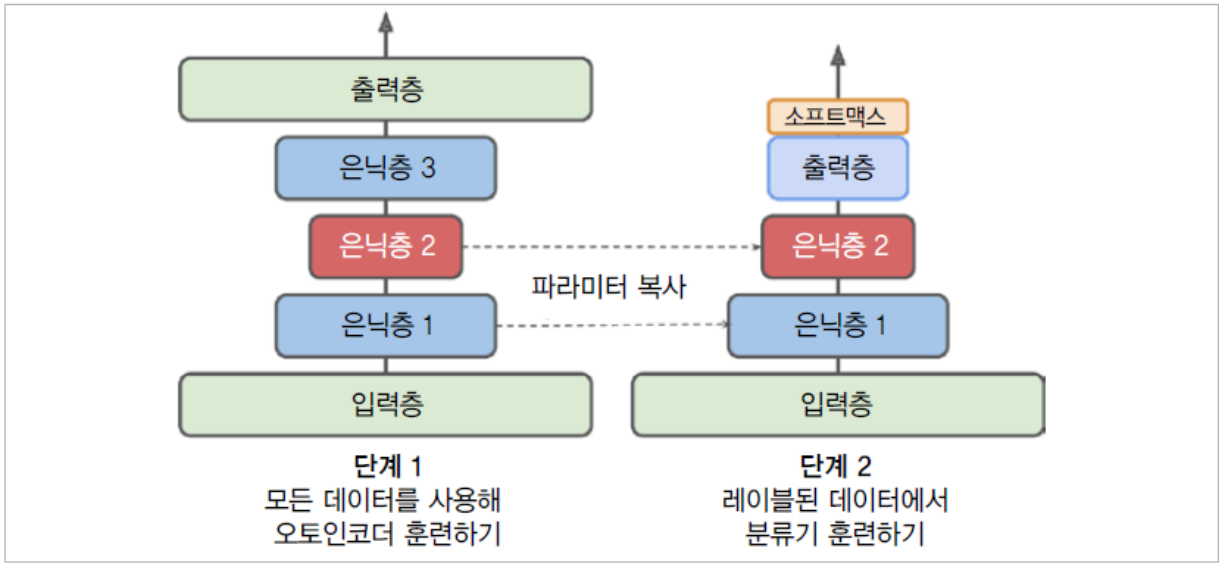


그림 17-6 오토인코더를 사용한 비지도 사전훈련

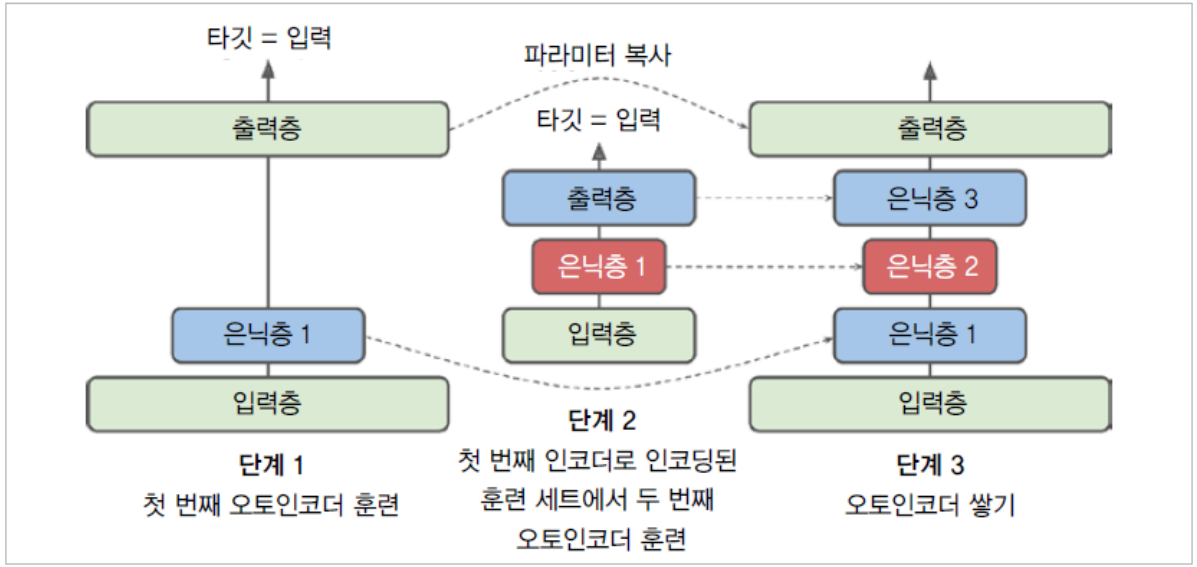


그림 17-7 한 번에 오토인코더 한 개씩 훈련하기

SECTION 17-4 합성곱 오토인코더

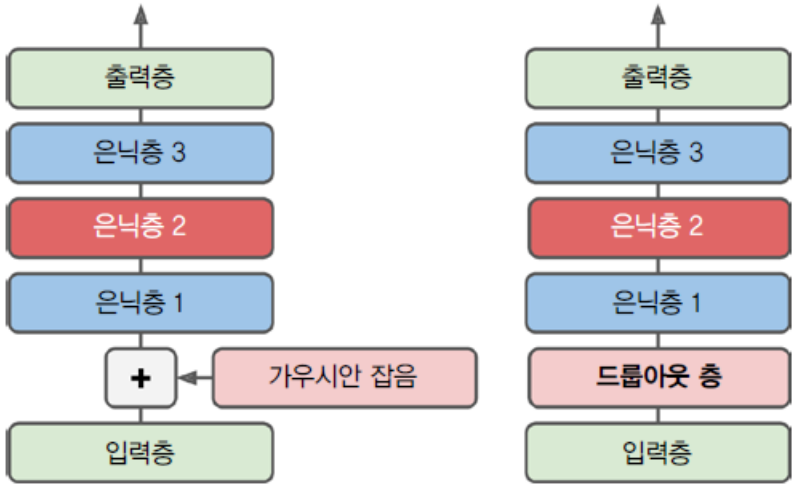
- 이미지를 다룰 때는 합성곱 신경망이 밀집 네트워크보다 훨씬 좋은 성능
  - 합성곱 오토인코더
    - 합성곱 층과 풀링 층으로 구성된 일반적인 CNN
    - 일반적인 인코더 입력: 공간 방향의 차원(즉, 높이와 너비)을 줄이고 깊이(즉, 특성 맵의 개수)를 늘림 디코더는 거꾸로 동작(이미지의 스케일을 늘리고 깊이를 원본 차원으로 되돌림)
    - 이를 위해서 전치 합성곱 층을 사용(또는 합성곱 층과 업샘플링 층을 연결).
    - 패션 MNIST 데이터셋에 대한 간단한 합성곱 오토인코더 학습

SECTION 17-5 순환 오토인코더

- 순환 오토인코더
  - (비지도 학습이나 차원 축소를 위해) 시계열이나 텍스트와 같은 시퀀스에 대한 오토인코더를 만들려면 순환 신경망이 밀집 네트워크보다 효과적
  - 인코더: 입력 시퀀스를 하나의 벡터로 압축하는 시퀀스-투-벡터 RNN
  - 디코더: 반대로 벡터-투-시퀀스 RNN
  - 각 이미지를 행의 시퀀스로 간주하여 처리
  - 과대완전 오토인코더

SECTION 17-6 잡음 제거 오토인코더

- 입력에 잡음을 추가하고, 잡음이 없는 원본 입력을 복원하도록 훈련
  - 데이터 시각화나 비지도 사전훈련을 위해 사용할 뿐만 아니라 간단하고 효율적으로 이미지에서 잡음을 제거하는 데 사용
  - 적층 잡음 제거 오토인코더
  - 잡음은 입력에 추가된 순수한 가우시안 잡음 또는 드롭아웃처럼 무작위로 입력을 꺼서 발생시킬 수도 있음



▲ 그림 17-8 가우시안 잡음(왼쪽) 또는 드롭아웃(오른쪽)을 사용한 잡음 제거 오토인코더

▲ 그림 17-9 잡음 섞인 이미지(위)와 재구성된 이미지(아래)

SECTION 17-7 희소 오토인코더

- 희소 오토인코더
  - 비용 함수에 적절한 항을 추가하여 오토인코더가 코딩 층에서 활성화되는 뉴런 수를 감소시키도록 작용
  - 예를 들어 코딩 층에서 평균적으로 5% 뉴런만 활성화되도록 강제하면, 오토인코더가 적은 수의 활성화된 뉴런을 조합하여 입력을 표현해야 함
  - 코딩 층에 시그모이드 활성화 함수를 활용, 큰 코딩 층(예를 들면 300개의 유닛을 가진 층)을 사용
  - 훈련 반복마다 코딩 층의 실제 희소 정도를 측정하고 타깃 희소 정도와 다르면 모델에 벌칙을 부과
  - KL 발산을 기반으로 하는 희소 오토인코더를 구현하기

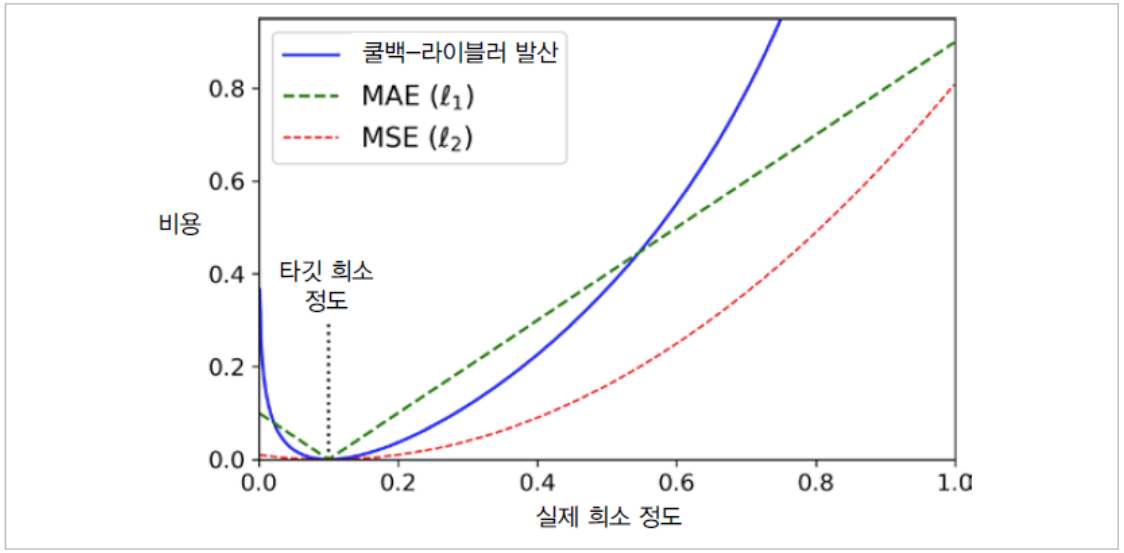
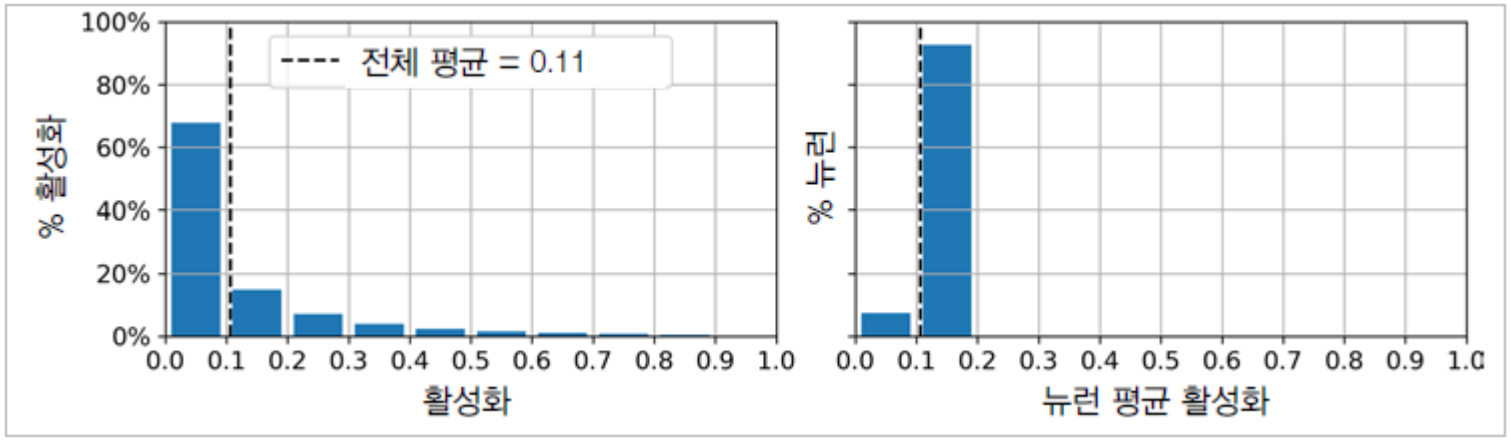


그림 17-10 희소 손실

SECTION 17-7 희소 오토인코더



▲ 그림 17-11 코딩 층에 있는 모든 활성화의 분포(왼쪽)와 각 뉴런의 평균 활성화 분포(오른쪽)

SECTION 17-8 변이형 오토인코더

- 변이형 오토인코더의 속성
  - 확률적 오토인코더: 즉, 훈련이 끝난 후에도 출력이 부분적으로 우연에 의해 결정(이와는 반대로 잡음 제거 오토인코더는 훈련 시에만 무작위성을 사용)
  - 생성 오토인코더: 마치 훈련 세트에서 샘플링된 것 같은 새로운 샘플을 생성할 수 있음
  - 입력이 복잡한 분포를 가지더라도 간단한 가우시안 분포에서 샘플링된 것처럼 보이는 코딩을 만드는 경향

- 패션 MNIST 이미지 생성하기
  - 가우시안 분포에서 랜덤한 코딩을 샘플링하여 디코딩
  - 변이형 오토인코더는 시맨틱 보간을 수행

식 17-3 변이형 오토인코더의 잠재 손실

$$\mathscr{L} = -\frac{1}{2}\sum_{i=1}^{n} 1 + \log\left(\sigma_i^{\,2}\right) - \sigma_i^{\,2} - \mu_i^{\,2}$$

식 17-4 $\gamma = \log(\sigma^2)$을 사용해 다시 쓴 변이형 오토인코더의 잠재 손실

$$\mathscr{L} = -\frac{1}{2}\sum_{i=1}^{n} 1 + \gamma_i - \exp\left(\gamma_i\right) - \mu_i^{\,2}$$

SECTION 17-8 변이형 오토인코더

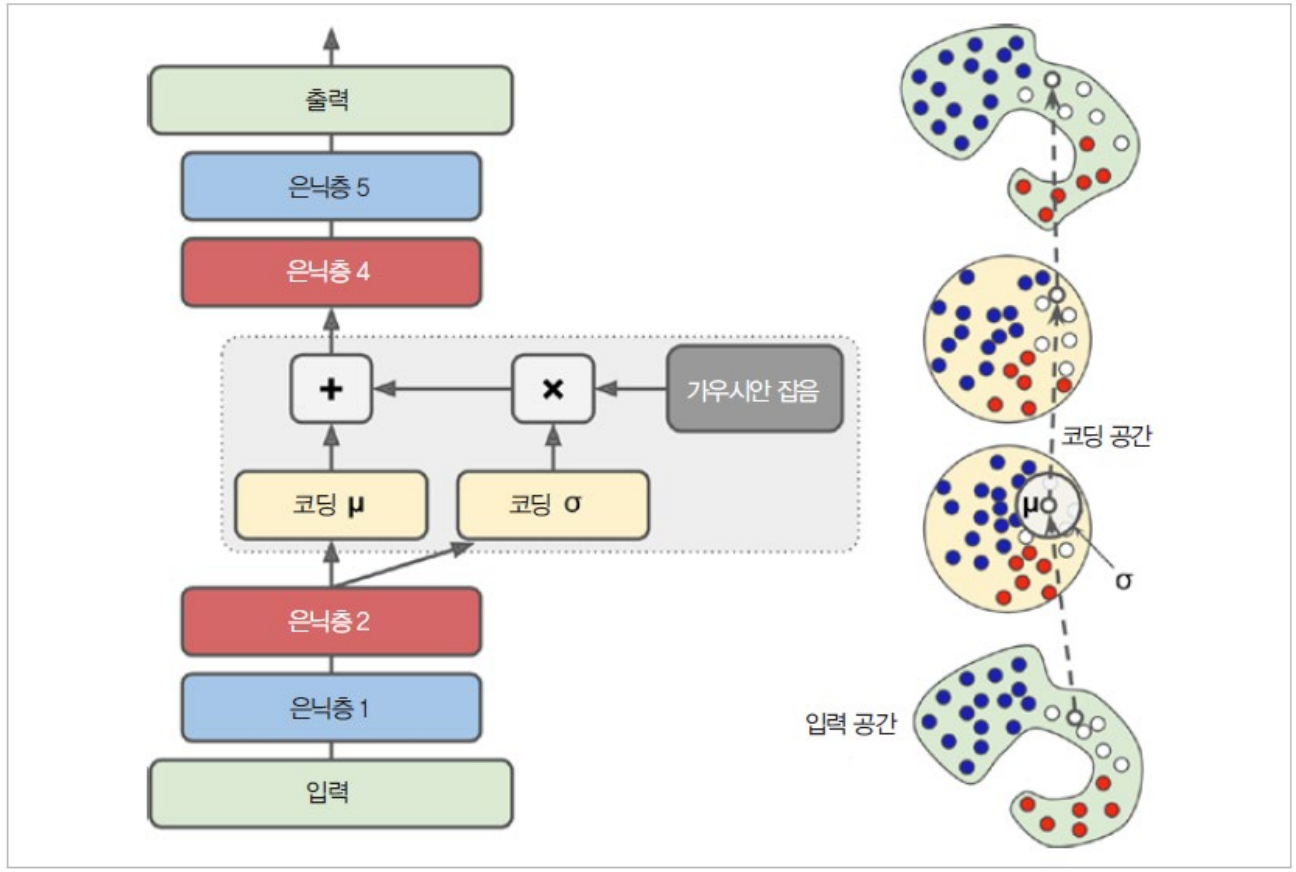


그림 17-12 변이형 오토인코더(왼쪽)와 이를 통과하는 샘플(오른쪽)

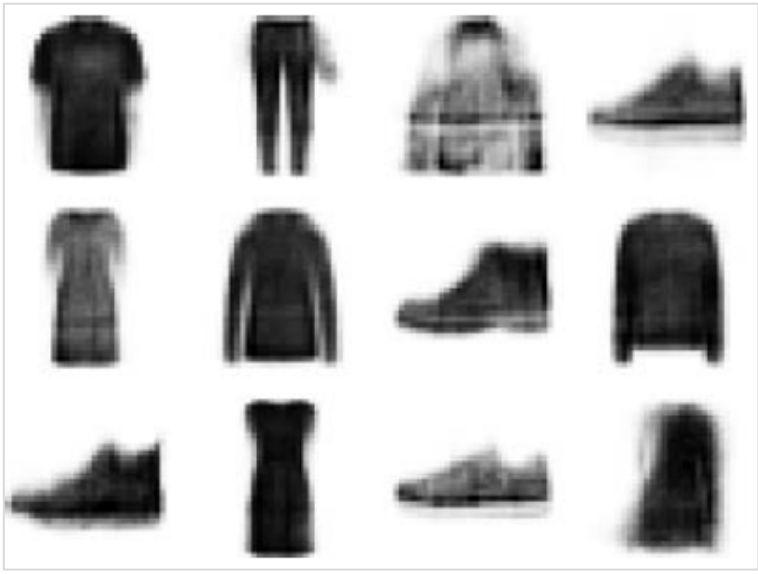
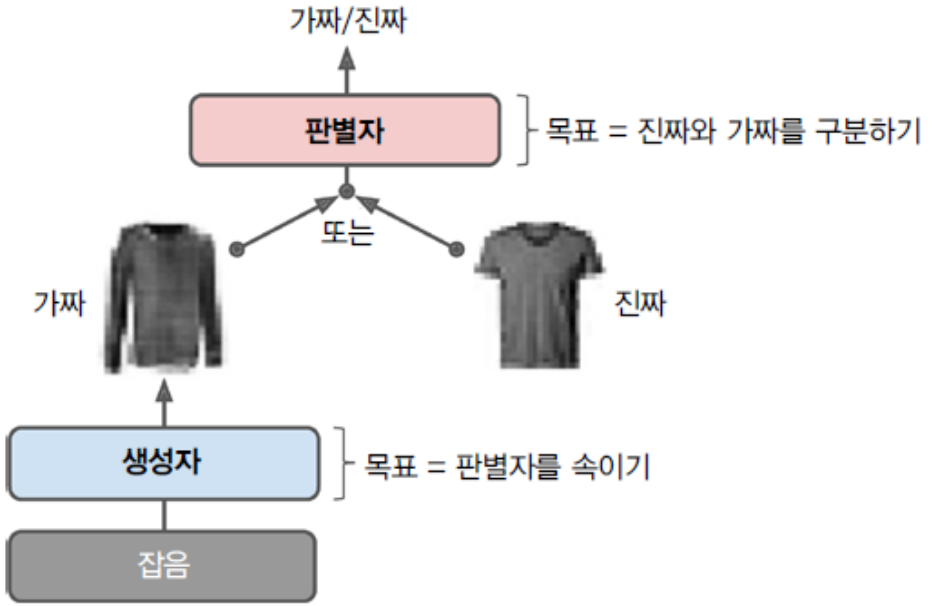


그림 17-13 변이형 오토인코더로 생성된 패션 MNIST 이미지

SECTION 17-9 생성적 적대 신경망

- 생성적 적대 신경망(GAN, Generative Adversarial Network)
  - 생성자: 랜덤한 분포(일반적으로 가우시안 분포)를 입력으로 받고 이미지와 같은 데이터를 출력
  - 판별자: 생성자에서 얻은 가짜 이미지나 훈련 세트에서 추출한 진짜 이미지를 입력으로 받아 입력된 이미지가 가짜인지 진짜인지 구분

- GAN 훈련의 어려움



▲ 그림 17-15 생성적 적대 신경망

SECTION 17-9 생성적 적대 신경망

그림 17-16 한 훈련 에포크가 끝난 후 GAN이 생성한 이미지

SECTION 17-9 생성적 적대 신경망

- 생성적 적대 신경망(GAN, Generative Adversarial Network)
  - **심층 합성곱 GAN(DCGAN)**
    안정적인 합성곱 GAN을 구축하기 위한 가이드라인
    - (판별자에 있는) 풀링 층을 스트라이드 합성곱으로 바꾸고 (생성자에 있는) 풀링 층은 전치 합성곱으로 바꾼다.
    - 생성자와 판별자에 배치 정규화를 사용한다. 생성자의 출력층과 판별자의 입력층은 제외한다.
    - 층을 깊게 쌓기 위해 완전 연결 은닉층을 제거한다.
    - tanh 함수를 사용해야 하는 출력층을 제외하고 생성자의 모든 층은 ReLU 활성화 함수를 사용한다.
    - 판별자의 모든 층은 LeakyReLU 활성화 함수를 사용한다.

그림 17-17 훈련 에포크 50번 후에 DCGAN이 생성한 이미지

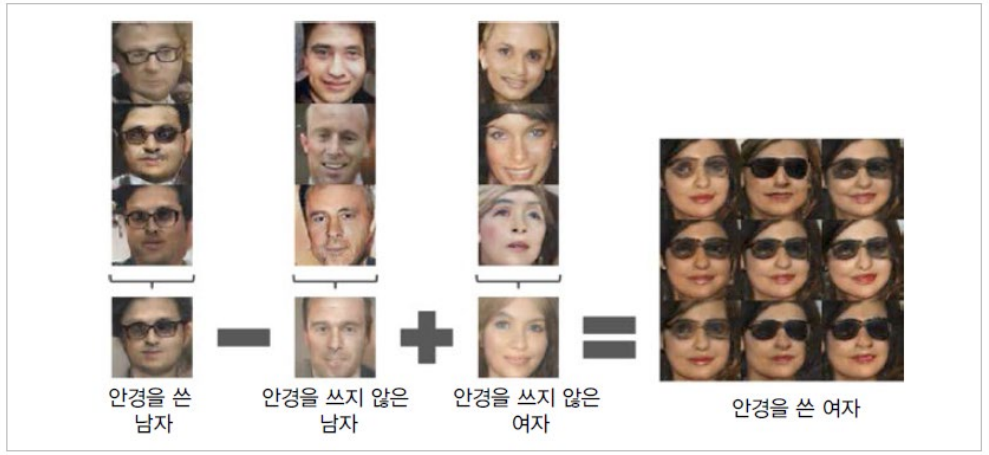


그림 17-18 시각적 개념의 벡터 연산(DCGAN 논문 Figure 7의 일부분)[24]

SECTION 17-9 생성적 적대 신경망

- 생성적 적대 신경망(GAN, Generative Adversarial Network)
  - ProGAN
    - 미니배치 표준편차 층
    - 동일한 학습 속도
    - 픽셀별 정규화 층

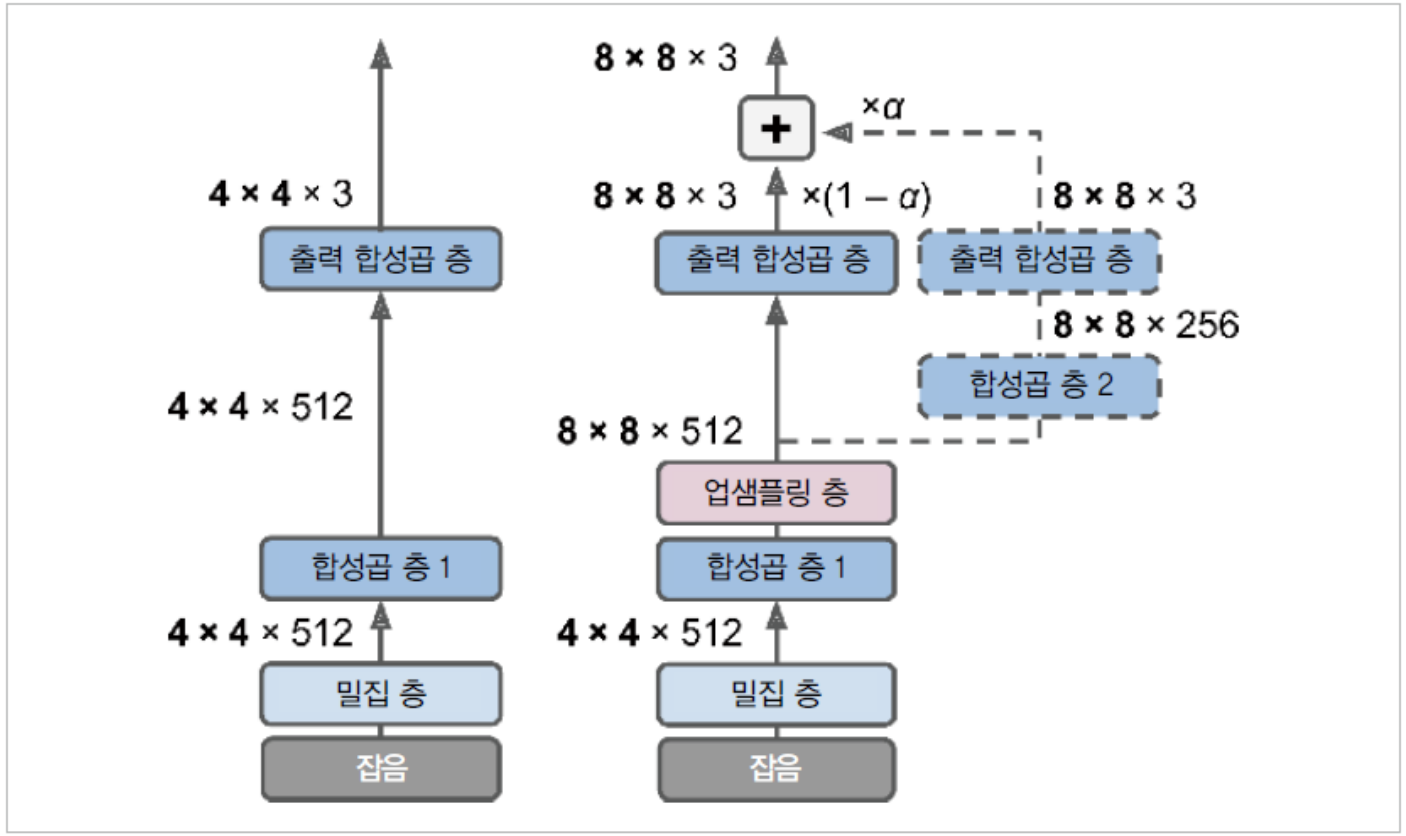


그림 17-19 ProGAN: GAN 생성자가 4×4 컬러 이미지를 출력합니다(왼쪽). 이를 8×8 이미지를 출력하도록 확장합니다(오른쪽).

SECTION 17-9 생성적 적대 신경망

- 생성적 적대 신경망(GAN, Generative Adversarial Network)
  - StyleGAN
    - 매핑 네트워크
    - 합성 네트워크

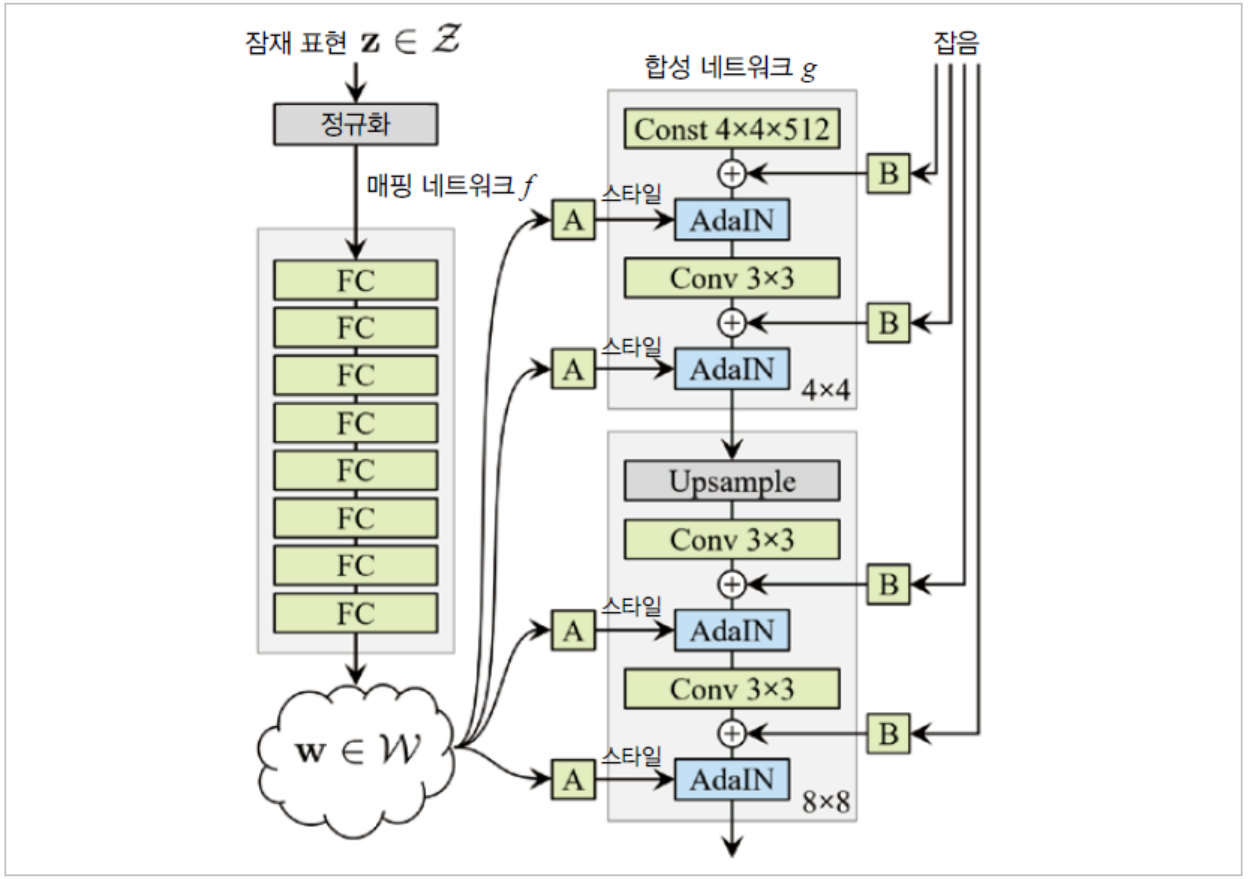


그림 17-20 StyleGAN의 생성자 구조(StyleGAN 논문에 있는 Figure 1의 일부분)[31]

SECTION 17-10 연습문제(1)

1. 오토인코더를 활용할 수 있는 주요 작업은 무엇인가?
2. 레이블되지 않은 훈련 데이터는 많지만, 레이블된 데이터는 수천 개 정도만 가지고 있을 때 분류기를 훈련하려 합니다. 오토인코더가 어떻게 도움이 될 수 있을까요? 어떻게 작업하면 될까?
3. 오토인코더가 완벽하게 입력을 재구성했다면, 이것이 반드시 좋은 오토인코더인가? 오토인코더의 성능을 어떻게 평가할 수 있나?
4. 과소완전과 과대완전 오토인코더가 무엇인가? 지나치게 과소완전인 오토인코더의 주요한 위험은 무엇인가? 과대완전 오토인코더의 주요한 위험은 무엇인가?
5. 적층 오토인코더의 가중치를 어떻게 묶나? 이렇게 하는 이유는 무엇인가?
6. 생성 모델이 무엇인가? 생성 오토인코더의 종류를 말할 수 있나?
7. GAN이 무엇인가? GAN이 유용한 몇 가지 작업을 나열할 수 있나?
8. GAN을 훈련할 때 주요 어려움은 무엇인가?

연습문제의 정답은 부록A에서 확인

SECTION 17-10 연습문제(2)

9. 잡음 제거 오토인코더를 사용해 이미지 분류기를 사전훈련해보기. (간단하게) MNIST를 사용하거나 도전적인 문제를 원한다면 CIFAR10 같은 좀 더 복잡한 이미지 데이터셋을 사용할 수 있음. 어떤 데이터셋을 사용하던지 다음 단계를 따라야 함
   - 데이터셋을 훈련 세트와 테스트 세트로 나누기. 전체 훈련 세트에서 심층 잡음 제거 오토인코더를 훈련.
   - 이미지가 잘 재구성되는지 확인. 코딩 층의 각 뉴런을 가장 크게 활성화하는 이미지를 시각화
   - 오토인코더의 아래 층을 재사용해 분류 DNN을 만들기. 훈련 세트에서 이미지 500개만 사용해 훈련. 사전훈련을 사용하는 것이 더 나은가? 사용하지 않는 것이 더 나은가?
10. 이미지 데이터셋을 하나 선택해 변이형 오토인코더를 훈련하고 이미지를 생성해보기. 또는 관심 있는 레이블이 없는 데이터셋을 찾아서 새로운 샘플을 생성할 수 있는지 확인해보기.
11. 이미지 데이터셋을 처리하는 DCGAN을 훈련하고 이를 사용해 이미지를 생성해보기. 경험 재생을 추가하고 도움이 되는지 확인. 생성된 클래스를 제어할 수 있는 조건 GAN으로 바꾸어 시도하기.

연습문제의 정답은 부록A에서 확인