# 17. 오토인코더와 GAN을 사용한 표현 학습과 생성적 학습

2021-05-30 백관구

# 오토인코더 (Autoencoder)

- **비지도(Unsupervised)** 방식으로 데이터의 **잠재 표현(latent representation)**을 **학습**하는 인공신경망

- 일반적으로, 입력데이터(고차원) → 밀집표현(저차원)

  ➔ **차원 축소**에 적용

- 잠재 표현 학습을 통해 훈련데이터와 유사한 데이터를 생성할 수 있음

  ➔ **생성** 모델

# 생성적 적대 신경망 (GAN; Generative Adversarial Networks)

- 이미지 : 고해상도, 채색, 변환(스케치→정교한 이미지), 다음 프레임 생성
- 데이터 : 훈련 샘플 증식

# 오토인코더 vs GAN

- 오토인코더
  - 입력을 출력으로 복사하는 방법을 학습 (쉬운데?)
  - 이 과정에서 여러 제약 조건(노이즈, 잠재표현 크기 제한)을 추가해 인공신경망이 **데이터의 효율적인 표현 방법을 학습**할 수 있도록 만듦

- GAN
  - **생성자, 판별자**로 구성
  - 생성자 : 훈련데이터와 유사한 샘플 생성
  - 판별자 : 실존하는 데이터인지, 생성된 데이터인지 구별
  - 생성자 vs 판별자 둘 다 학습시킴 ➔ **적대적 학습**

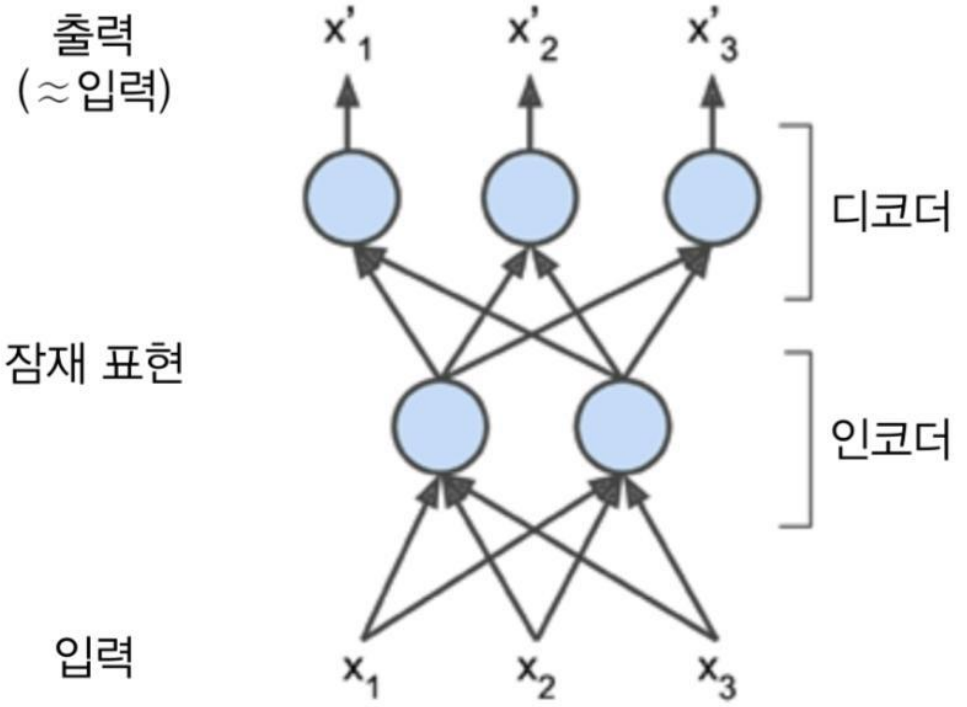
# 17.1. 효율적인 데이터 표현 (오토인코더)

- 0 27 25 36 81 51 10 83 73 19
- 0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
- 인간이 위 두 배열을 기억해서 받아 적는다고 하면,
  - 첫 번째 배열이 두 번째 배열보다 길이는 짧지만,
  - 두 번째 배열의 규칙(패턴)을 알고 있으므로 기억하기는 더 쉬움!
- 오토인코더는 데이터의 숨겨진 패턴, 즉 잠재표현을 찾아 **데이터를 효율적으로 표현하는 방법을 학습**

# 17.1. 효율적인 데이터 표현

- 인코더(Encoder)
  - 숫자 배열 → 잠재표현(패턴)
- 디코더(Decoder)
  - 잠재표현 → 학습한 숫자 배열
- 인코더-디코더를 통과해 재구성한 생성된 데이터는 입력데이터(원본)와 완전 똑같지 않음 (제약조건 때문에)
- **재구성 손실** : 생성데이터와 입력데이터의 차이

  ➔ 비용함수



# 17.2. 과소완전 선형 오토인코더로 PCA

- 과소완전 : 잠재표현의 차원이 입력데이터의 차원보다 작은 경우
- 선형 오토인코더 : 선형 활성화함수만 사용

```
from tensorflow import keras

encoder = keras.models.Sequential([keras.layers.Dense(2, input_shape=[3])])
decoder = keras.models.Sequential([keras.layers.Dense(3, input_shape=[2])])
autoencoder = keras.models.Sequential([encoder, decoder])

autoencoder.compile(loss="mse", optimizer=keras.optimizers.SGD(lr=0.1))
```

```
history = autoencoder.fit(X_train, X_train, epochs=20)
codings = encoder.predict(X_train)
```

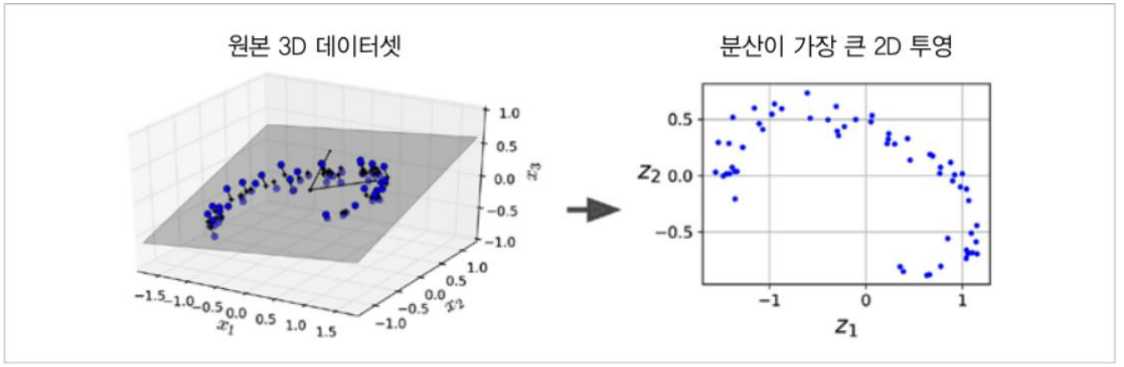



그림 17-2 과소완전 선형 오토인코더로 수행한 PCA