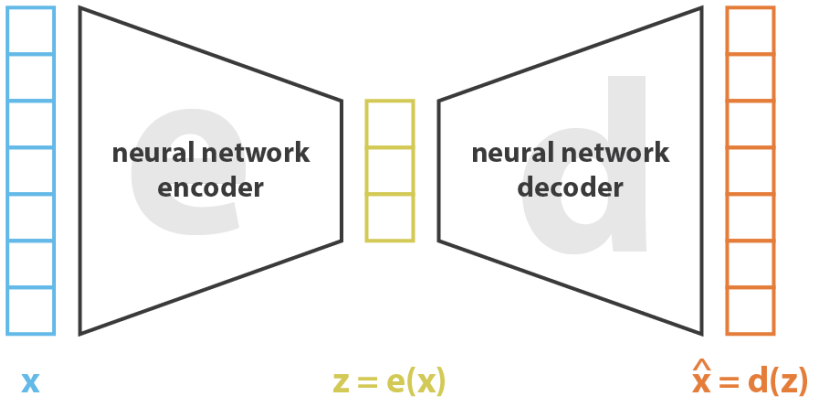
Autoencoder

1. Autoencoder란?



$$loss = || x - \hat{x} ||^2 = || x - d(z) ||^2 = || x - d(e(x)) ||^2$$

특징

- 입출력이 동일한 네트워크
  - 즉, Loss는 ouput이 input과 동일하도록 장려
- Encoder : 훈련 데이터를 latent vector로 표현
- Decoder : latent vector를 다시 훈련 데이터로 표현

활용

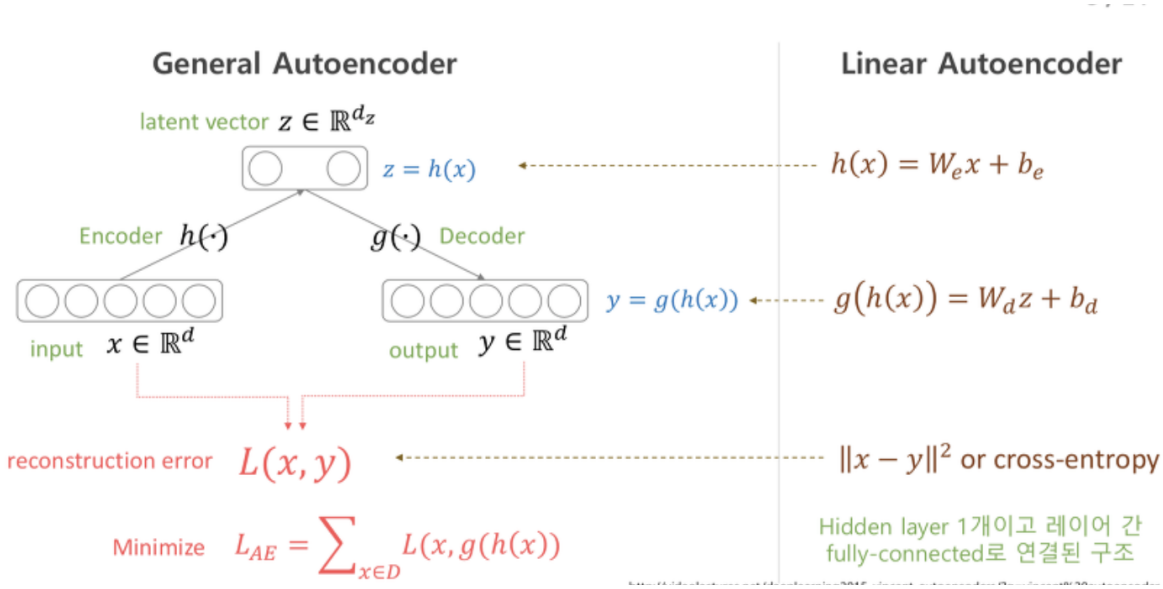
- 차원 축소에 활용할 수 있음
  1. Supervised Learning을 통한 차원 축소
  2. 비선형성

Linear AutoEncoder

- 만약 선형적으로 차원 축소를 하고 싶다면, activation function을 사용하지 않으면 됨

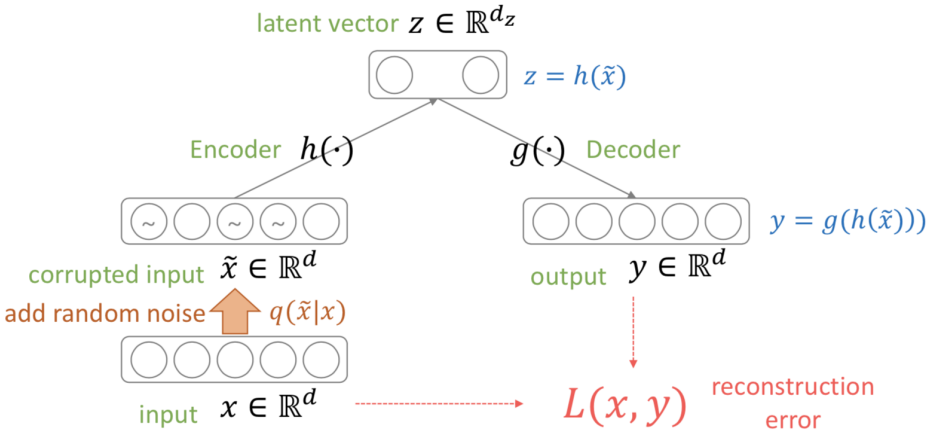
- linear autoencoder를 구성하고, loss function을 MSE를 사용하면 PCA와 똑같은 manifold를 학습함(=weight가 같은 subspace를 span함) (basis는 다를 수 있음)
    - 하지만 PCA가 더 효율적임

$$Minimize\ L_{AE} = \sum_{x \in D} L(x, g(h(x)))$$



2. 다양한 AutoEncoders

1) Denosing AutoEncoder (DAE)



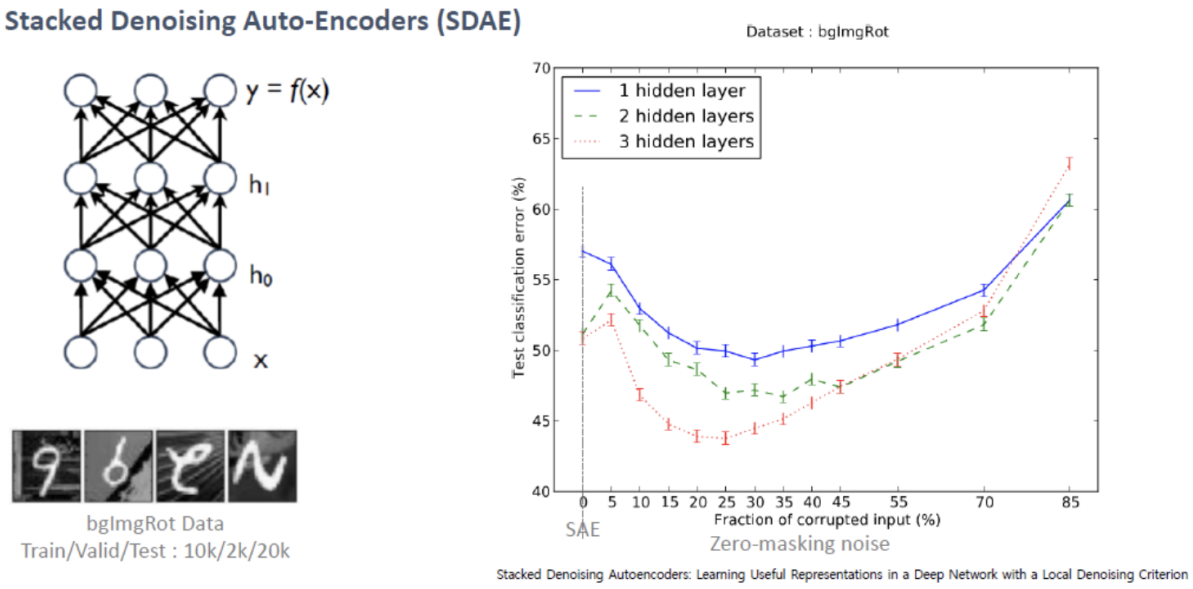
$$Minimize\ L_{DAE} = \sum_{x \in D} E_{q(\tilde{x}|x)} [L(x, g(h(\tilde{x})))]$$

- input에 noise를 추가하고, noise가 없는 input을 복원하도록 훈련하는 것
- noise가 추가된 input은 원 데이터는 input과 다르지만, 의미적으로는 같기 때문에 같은 차원축소 표현을 가지게 됨

$$Minimize\ L_{DAE} = \sum_{x \in D} E_{q(\tilde{x}|x)}[L(x, g(h(\tilde{x}))]$$

- 잡음 추가 방법 $\tilde{x} \sim q(\tilde{x}|x)$
  1. 마스킹 잡음
     - dropout처럼 무작위로 입력을 끔으로써(=입력값을 0으로 만듦으로써) 발생시킬 수 있음
     - 0으로 만드는 비율을 너무 높이면 성능이 나빠짐

       (약 25%의 input에 대해 noise를 추가해주었을 때 가장 loss가 낮음)
  2. 가우스 잡음



2) Stochastic Constractive AutoEndoer (SCAE)

- 목적함수에 정규화항을 추가하여, 모델이 입력값의 약간의 변형에 대해 덜 민감하도록 학습

$$L_{SACE} = \sum_{x \in D} L(x, g(h(x))) + \lambda E_{q(\tilde{x}|x)}[||h(x) - h(\tilde{x})||^2]$$

3. 다양한 Neural Network 구성

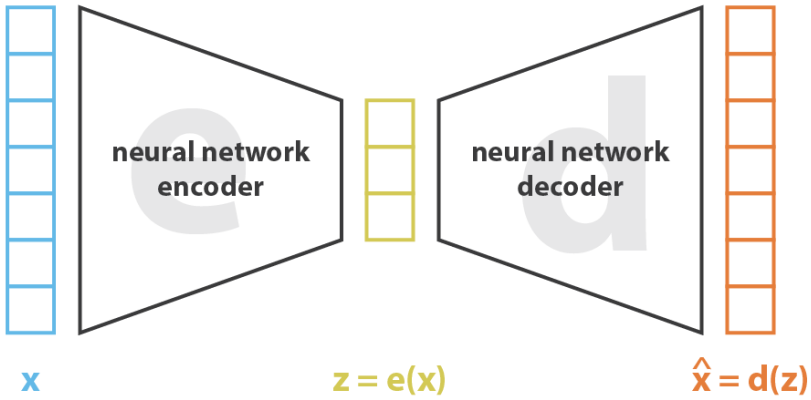
1) CNN으로 구성

- 이미지를 다룰 때는 Dense layer 외에 CNN을 통해 오토인코더를 구성할 수 있음
- Encoder
  - 합성곱과 풀링층으로 구성된 일반적인 CNN
- Decoder
  - 전치 합성곱 층 사용

2) RNN으로 구성

- 시계열/텍스트와 같은 시퀀스를 다룰 땐, RNN 사용
- Encoder
  - 입력 시퀀스를 하나의 벡터로 압축
- Decoder
  - 하나의 벡터를 입력 시퀀스로 복원

4. Variational Autoencoders

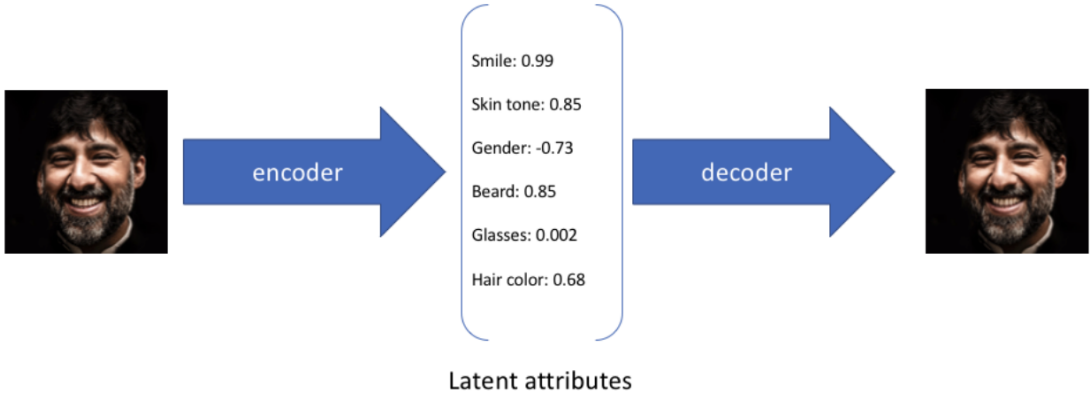


$$\text{loss} = \|x - \hat{x}\|^2 = \|x - d(z)\|^2 = \|x - d(e(x))\|^2$$

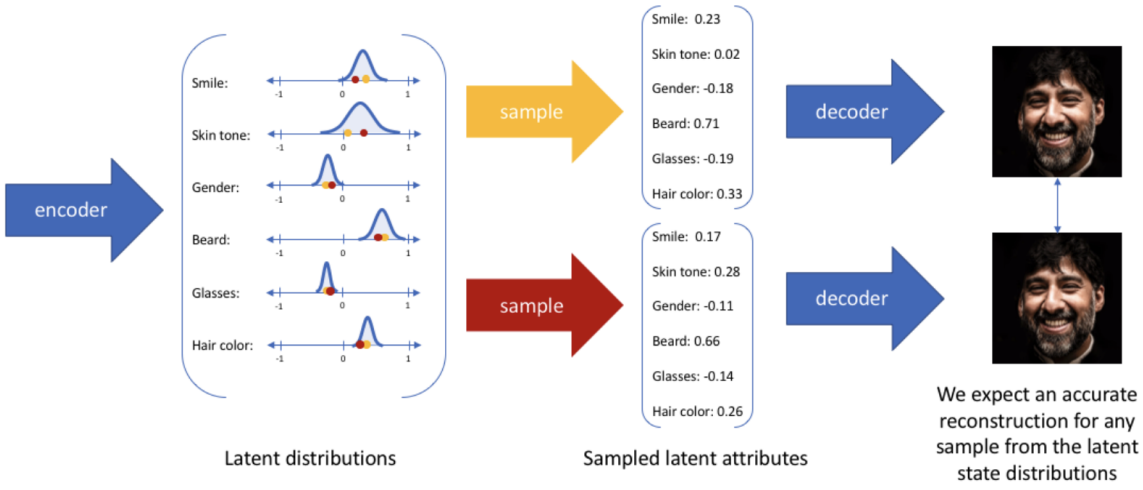
- 오토인코더가 차원축소의 목적(manifold learning)이라면 VAE는 데이터를 생성하는 목적(generative model)
- 즉, Decoder를 위해서 Encoder가 생겨남

AE vs VAE

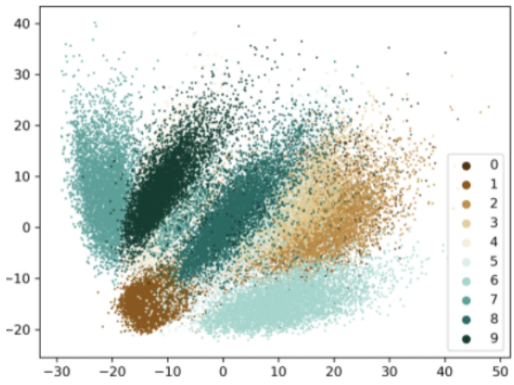
- Autoencoder
  - input data가 encoder를 통해 하나의 encoding vector로 변형됨
  - 즉, latent vector에 대한 하나의 single value를 출력



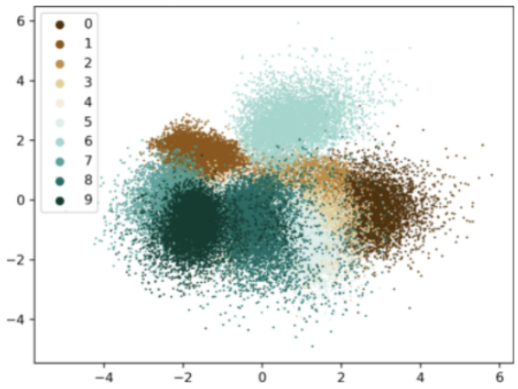
- VAE
  - encoder는 각 latent attribute에 대한 확률 분포를 출력
  - 해당 확률 분포에서 랜덤 샘플링된 값 z을 통해 이미지 생성



- 성능 차이
  - AE에 의해 생성된 latent distribution은 VAE에 의해 생성된 latent distribution에 비해 멀리 퍼져있음(이산적임)
  - 따라서 데이터를 재구성할 때 데이터의 품질이 낮아짐

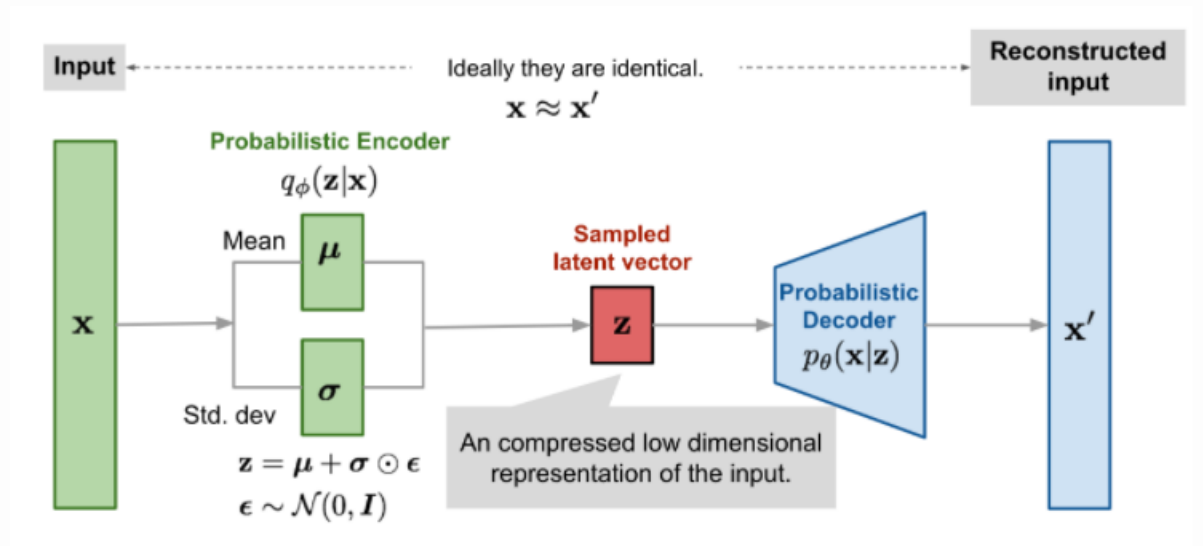


(a) *Latent Distribution by Label for AE*



(b) *Latent Distribution by Label for VAE*

구조



- 목적 : 데이터 샘플 x가 나올 likelihood($p_\theta(x)$)를 최대화하는 방향으로 학습

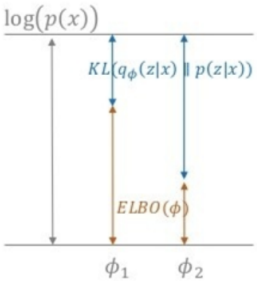
$$\log(p(x)) = \int \log(p(x)) q_\phi(z|x) dz \quad \leftarrow \int q_\phi(z|x)dz = 1$$

$$= \int \log\left(\frac{p(x,z)}{p(z|x)}\right) q_\phi(z|x)dz \quad \leftarrow p(x) = \frac{p(x,z)}{p(z|x)}$$

$$= \int \log\left(\frac{p(x,z)}{q_\phi(z|x)} \cdot \frac{q_\phi(z|x)}{p(z|x)}\right) q_\phi(z|x)dz$$

$$= \underbrace{\int \log\left(\frac{p(x,z)}{q_\phi(z|x)}\right) q_\phi(z|x)dz}_{ELBO(\phi)} + \underbrace{\int \log\left(\frac{q_\phi(z|x)}{p(z|x)}\right) q_\phi(z|x)dz}_{KL\left(q_\phi(z|x) \,\|\, p(z|x)\right)}$$

두 확률분포 간의 거리≥0



KL을 최소화하는 $q_\phi(z|x)$의 ϕ값을 찾으면 되는데 $p(z|x)$를 모르기 때문에,
KL최소화 대신에 ELBO를 최대화하는ϕ값을 찾는다.

$$ELBO(\phi) = \int \log\left(\frac{p(x,z)}{q_\phi(z|x)}\right) q_\phi(z|x)dz$$

$$= \int \log\left(\frac{p(x|z)p(z)}{q_\phi(z|x)}\right) q_\phi(z|x)dz$$

$$= \int \log(p(x|z)) q_\phi(z|x)dz - \int \log\left(\frac{q_\phi(z|x)}{p(z)}\right) q_\phi(z|x)dz$$

$$= \mathbb{E}_{q_\phi(z|x)}[\log(p(x|z))] - KL(q_\phi(z|x)||p(z))$$ 앞 슬라이드에서의 KL과 인자가 다른 것에 유의

$$logp(x) \geq E_{z\sim q(z|x)}[logp(x|z)] - D_{KL}(q(z|x)||p(z)) = ELBO$$

$$argmin\, L = -E_{z\sim q(z|x)}[logp(x|z)] + D_{KL}(q(z|x)||p(z))$$

- 첫번째 항은 reconstruction loss
- 두번째 항은 KL Divergence Regularizer
  - $z \sim N(0,1)$
  - $q(z|x) \sim N(u_{q(x)}, \sigma_{q(x)})$

$$-\frac{1}{2}\sum_{i=1}^{m} 1 + log(\sigma_i^2) - \sigma_i^2 - \mu_i^2$$