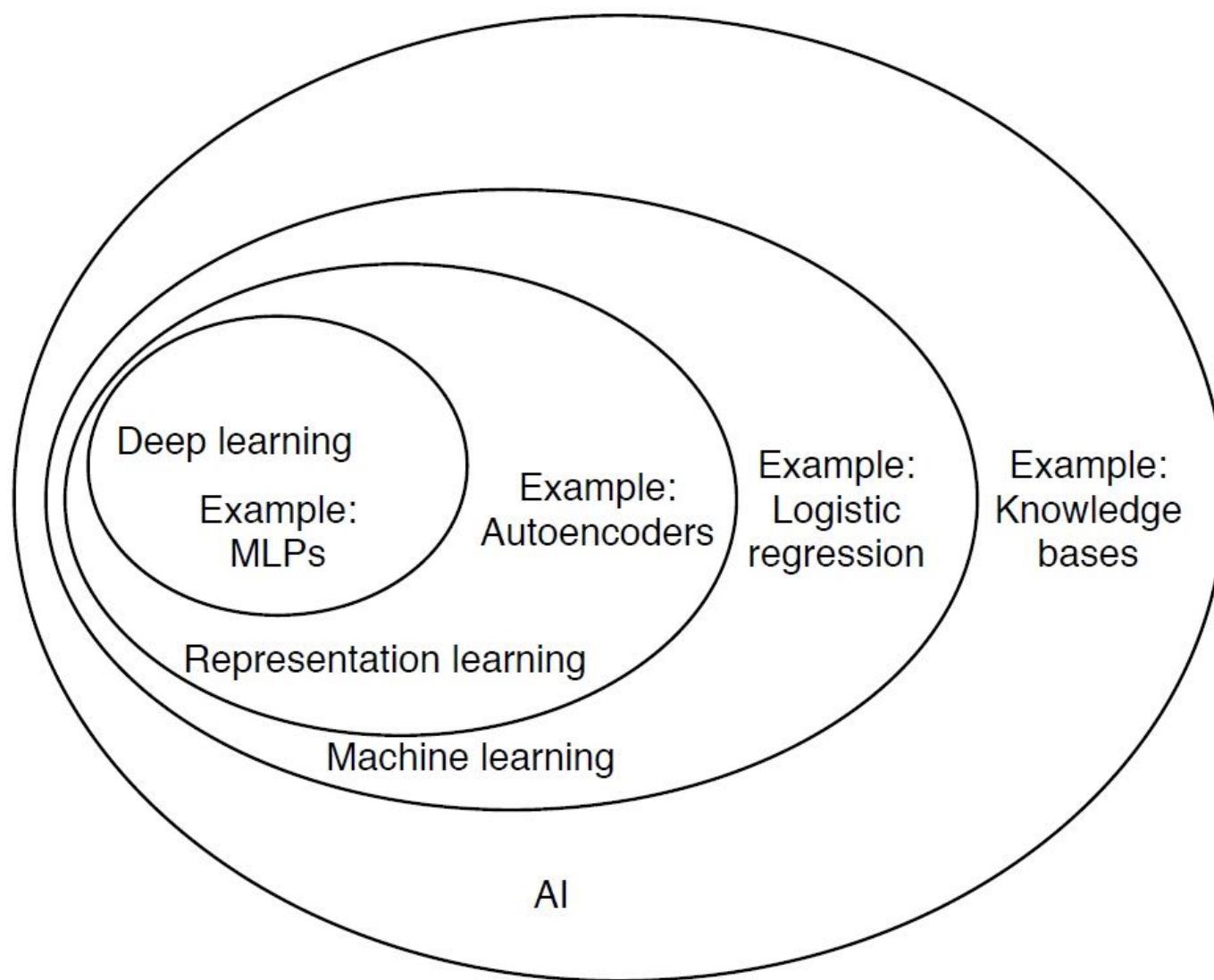
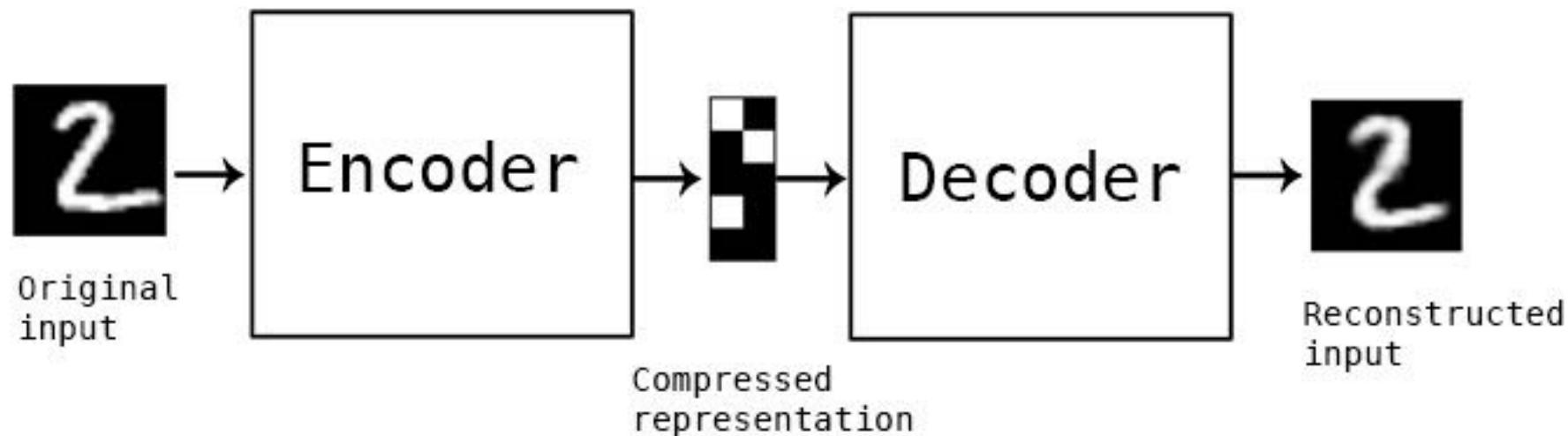


# AutoEncoder



Data Specific

Loss



- Unsupervised learning
- Nonlinear Dimensionality reduction
  - = Representation learning
  - = Efficient coding learning
  - = Feature extraction
  - = Manifold learning
- Generative model learning

# Autoencoder 3 요소

인코딩 함수 (encoding function)

디코딩 함수 (decoding function)

원본에 대해 압축된 표현(representation)과 압축 해제된  
표현(representation) 간 정보 손실량 간의 거리 함수 (즉, 손실 함수)

autoencoder가 비지도 학습(unsupervised learning)의 문제를 풀어낼 잠재적인 수단

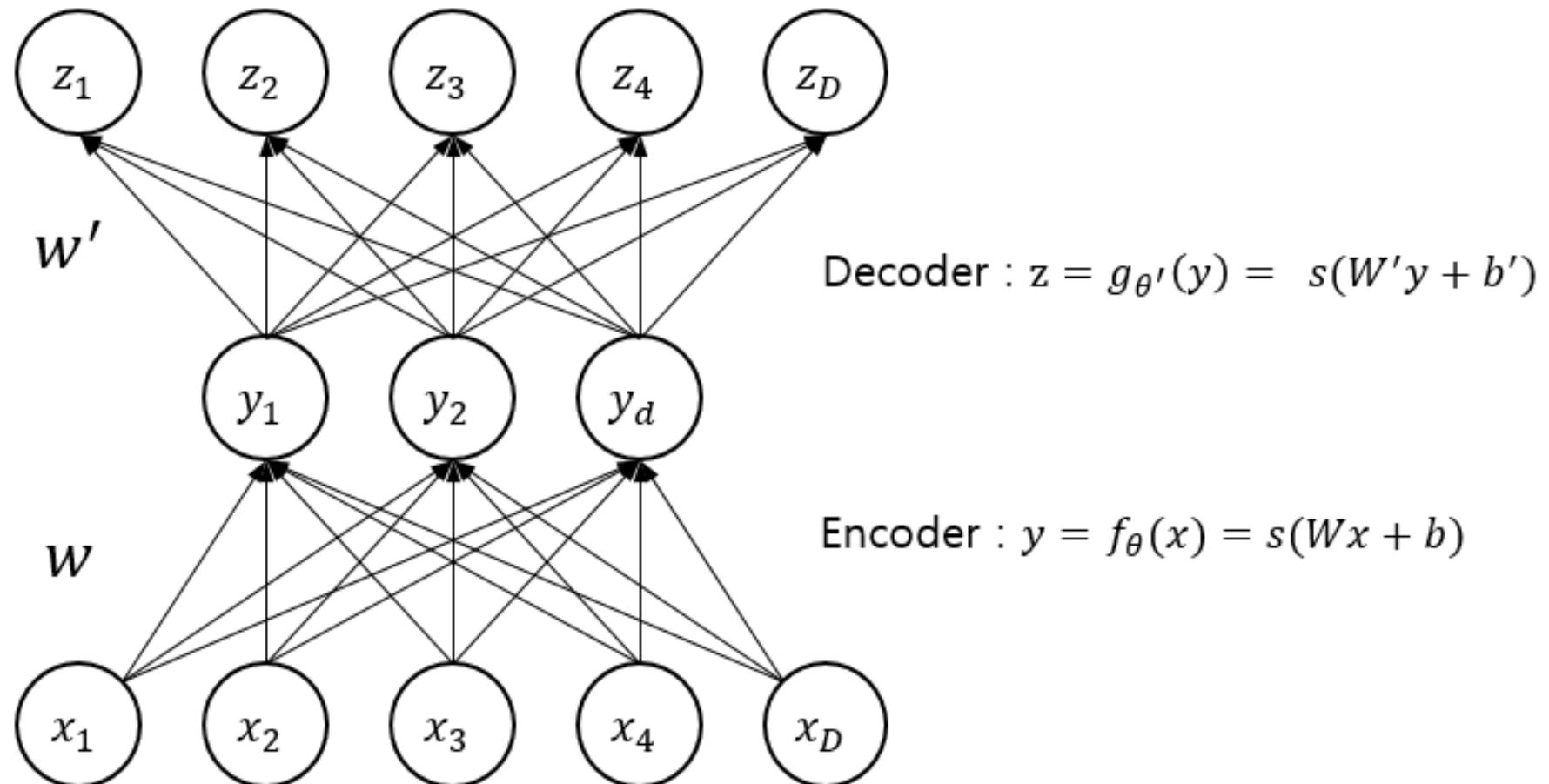
But autoencoder는 진정한 비지도 학습(unsupervised learning) 기술이 아니라, self-supervised 기술

지도 학습(supervised learning)의 일종으로 입력 데이터로부터 타겟을 만들어 냄  
흥미로운 특징(feature)들을 학습하는 self-supervised model을 얻으려면 흥미로운  
합성 목표 및 손실 함수를 제공

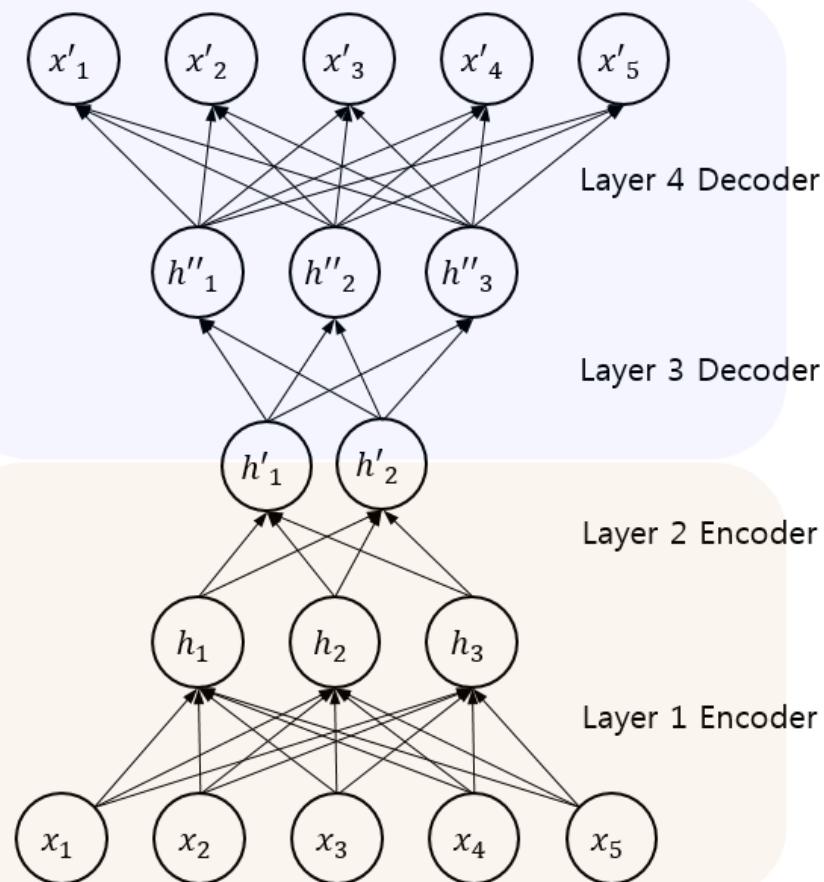
# Autoencoder 활용

- data denosing
- 데이터 시각화를 위한 차원 축소
  - 적절한 dimensionality와 sparsity constraints를 사용
    - autoencoder는 PCA나 다른 기법들보다 더 흥미로운 data projection을 학습
  - t-SNE(상대적으로 낮은 차원의 데이터를 요구)를 통한 2차원 시각화
    - autoencoder를 사용하여 데이터를 낮은 차원으로 압축

## Auto-Encoder (Basic form)



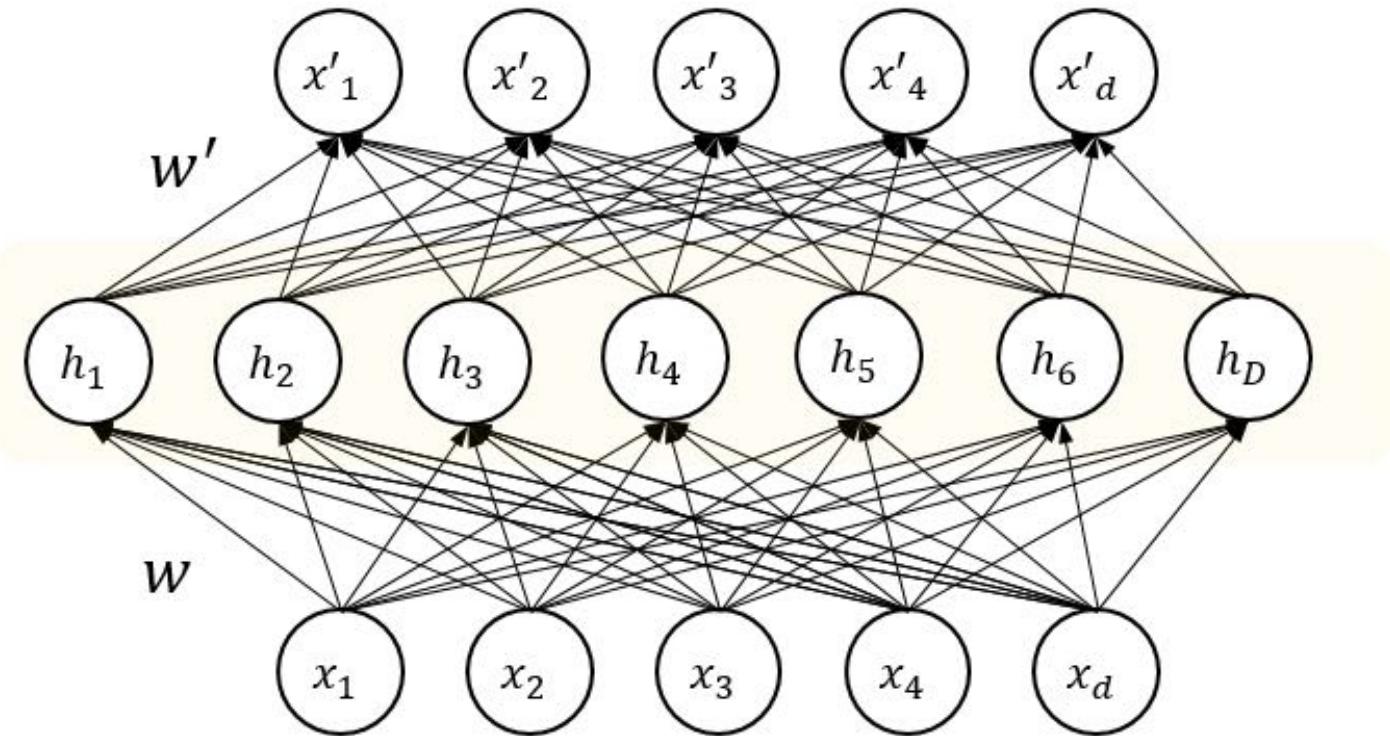
# Stacked Auto-Encoder



**Deep Generative Model** 이라고 할 수 없는 것  
 RBM에서는 probabilistic에 의존하여, data만 가지고도 label 을 관측하려 하는데, SAE에서는 사실상 deterministic 방법으로 모델을 학습

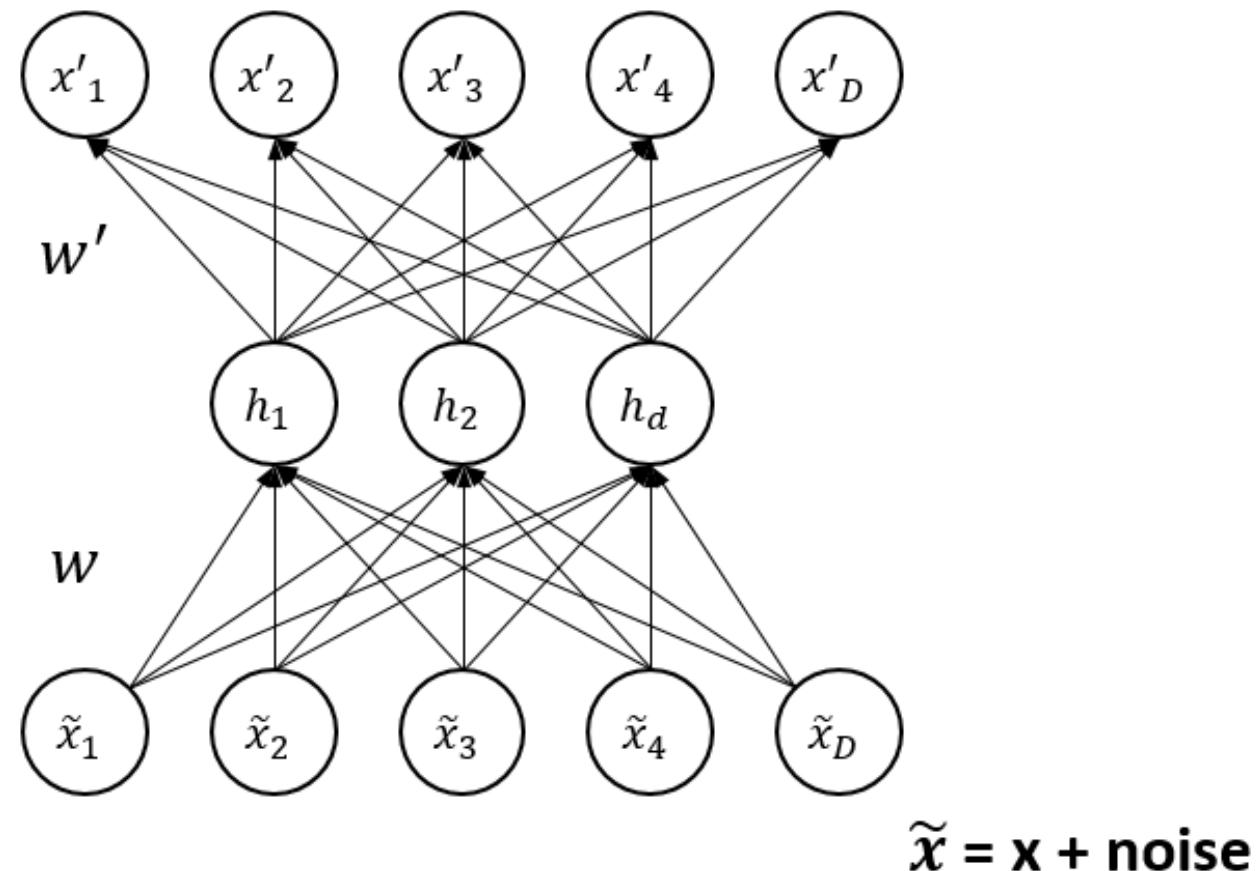
단점으로는 완벽한 Deep Generative Model 은 아니라는 것이지만, 장점은 training 에 속도가 빠르고, Deep Neural Nets 의 속성들을 이용할 수 있다는 것

## Sparse Auto-Encoder



히든 유닛의 수가 Input 데이터의 차원의 수(Input unit의 수)에 비해 상당히 많은 경우에 Sparsity Parameter을 제어하여, Hidden Unit의 Activation을 제어

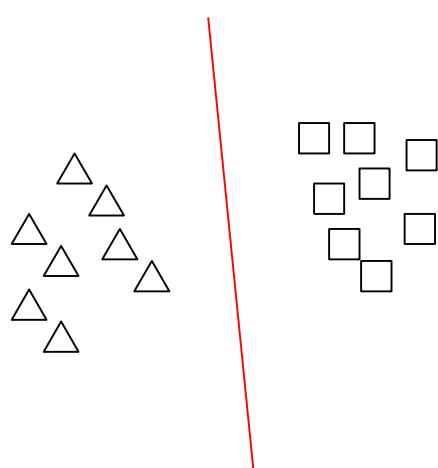
## Denoising Auto-Encoder (dA)



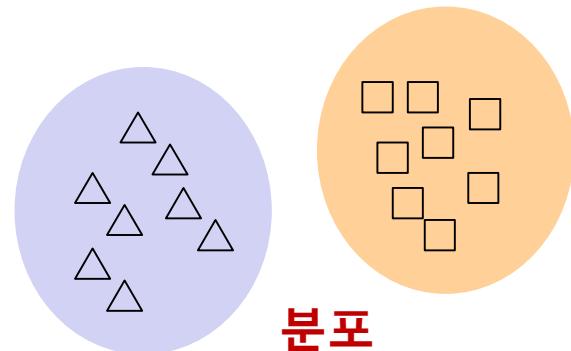
# Variational AutoEncoder

- 일반적인 기계학습은 판별모델
  - 각각을 나누기 위해 선을 긋는다 !
- 생성모델은 판별하는 것이 아니라 범위를 고려

판별모델



생성모델



$$p(x|z) = \frac{p(x \wedge z)}{p(z)} = \frac{p(z|x)p(x)}{p(z)}$$

Bayesian Probability은 사후확률(posterior probability)을 사전확률(prior probability)과 likelihood를 이용해서 계산할 수 있도록 해 주는 확률 변환식

- ☞ **likelihood**:  $p(z|x)$ , 어떤 모델에서 해당 데이터(관측값)이 나올 확률
- ☞ **사전확률(prior probability)**:  $p(x)$ , 관측자가 관측을 하기 전에 시스템 또는 모델에 대해 가지고 있는 선형적 확률. 예를 들어, 남여의 구성비를 나타내는  $p(\text{남자})$ ,  $p(\text{여자})$  등이 사전확률에 해당한다.
- ☞ **사후확률(posterior probability)**:  $p(x|z)$ , 사건이 발생한 후(관측이 진행된 후) 그 사건이 특정 모델에서 발생했을 확률

확률을 이용해서 classification 문제를 푸는 방법은 크게 2가지

### ML(Maximum Likelihood) 방법과 MAP(Maximum A Posteriori)

관측값을  $z$ , 그 값이 나온 클래스(또는 모델)를  $x$ 라 하자. 예를 들어, 바닥에 떨어진 머리카락의 길이( $z$ )를 보고 그 머리카락이 남자 것인지 여자 것인지 성별( $x$ )을 판단하는 문제를 생각해 보자.

• **ML(Maximum Likelihood) 방법:** ML 방법은 남자에게서 그러한 머리카락이 나올 확률  $p(z|\text{남})$ 과 여자에게서 그러한 머리카락이 나올 확률  $p(z|\text{여})$ 을 비교해서 가장 확률이 큰, 즉 likelihood가 가장 큰 클래스(성별)를 선택하는 방법이다.

• **MAP(Maximum A Posteriori) 방법:** MAP 방법은  $z$ 라는 머리카락이 발견되었는데 그것이 남자것일 확률  $p(\text{남}|z)$ , 그것이 여자것일 확률  $p(\text{여}|z)$ 를 비교해서 둘 중 큰 값을 갖는 클래스(성별)를 선택하는 방법이다. 즉, 사후확률(posterior probability)을 최대화시키는 방법으로서 MAP에서 사후확률을 계산할 때 Bayes 정리가 이용

예) 인구의 90%가 남자고 여자는 10% 밖에 없음

- ML은 남녀의 성비는 완전히 무시하고 순수하게 남자중에서 해당 길이의 머리카락을 가질 확률, 여자중에서 해당 길이의 머리카락을 가질 확률만을 비교
- MAP는 각각의 성에서 해당 머리카락이 나올 확률 뿐만 아니라 남녀의 성비까지 고려하여 최종 클래스를 결정

$$p(\text{여}|z) = \frac{p(\text{여} \wedge z)}{p(z)} = \frac{p(\text{여} \wedge z)}{p(\text{여} \wedge z) + p(\text{남} \wedge z)} = \frac{p(z|\text{여})p(\text{여})}{p(z|\text{여})p(\text{여}) + p(z|\text{남})p(\text{남})}$$

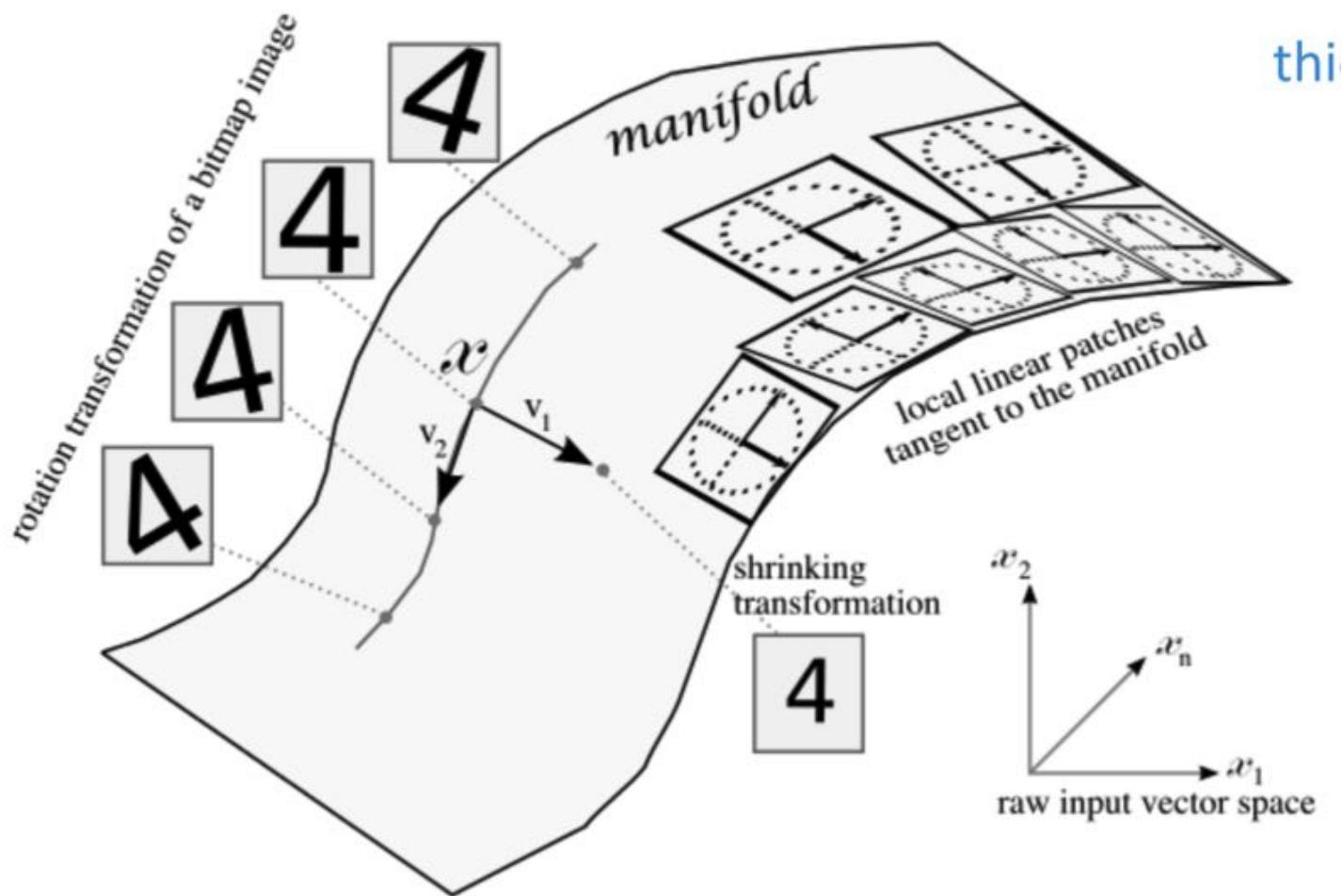
ML보다는 MAP 방법이 보다 정확한 classification 방법임을 알 수 있다. 하지만 많은 경우, 사전확률(prior probability)인  $p(\text{남})$ ,  $p(\text{여})$ 를 모르는 경우가 대부분이기 때문에 단순하게  $p(\text{남}) = p(\text{여})$ 로 놓고 문제를 푸는 경우가 많은데, 이 경우 MAP는 ML과 같게 됨

# Manifold

- 데이터가 있는 공간
  - 데이터는 다양한 차원에서 존재할 수 있는데, 이 차원을 축소시켜서 작은 차원으로 데이터를 보게 되면, 필요한 매개변수가 적어져 분류가 쉬워질 수 있음.

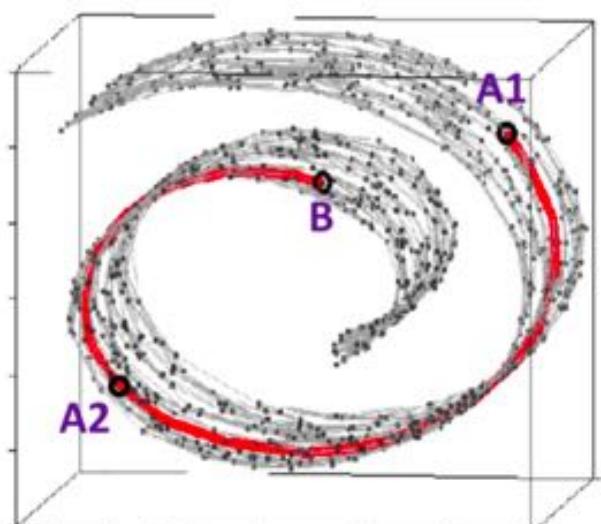
# Manifold Hypothesis

- 데이터가 고차원이라도 이 집합을 포함하는 저차원의 manifold 가 있음
  - 즉, 데이터가 고차원이라도 저차원의 manifold 상에 위치할 수 있으며, 그 낮은 차원의 manifold를 벗어나면 밀도가 낮아진다는 것.

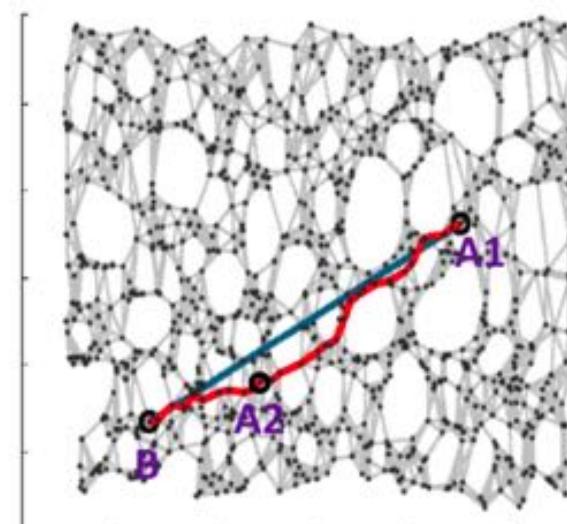


## Reasonable distance metric

의미적으로 가깝다고 생각되는 고차원 공간에서의 두 샘플들 간의 거리는 먼 경우가 많다.  
 고차원 공간에서 가까운 두 샘플들은 의미적으로는 굉장히 다를 수 있다.  
 차원의 저주로 인해 고차원에서의 유의미한 거리 측정 방식을 찾기 어렵다.



Distance in high dimension



Distance in manifold

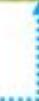
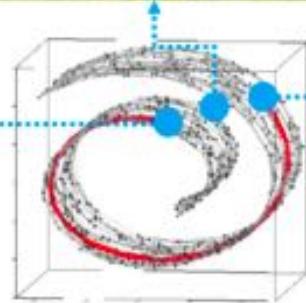
중요한 특징들을  
찾았다면 이 특징을  
공유하는 샘플들도  
찾을 수 있어야 한다.



## Reasonable distance metric

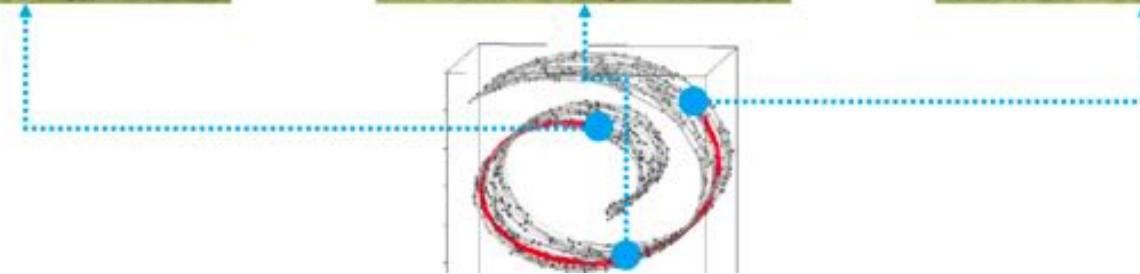


Interpolation in high dimension

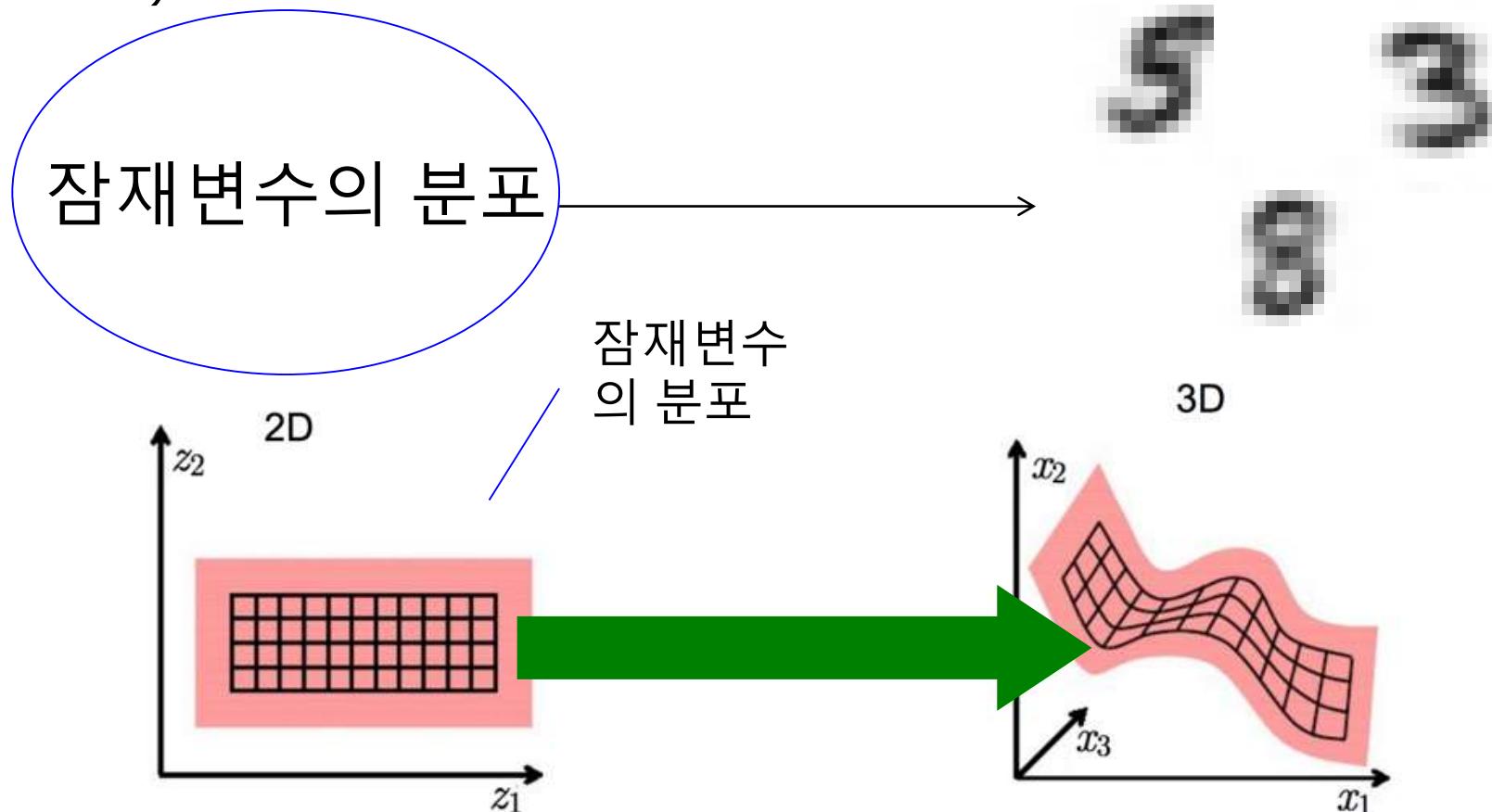


## Reasonable distance metric

Interpolation in manifold

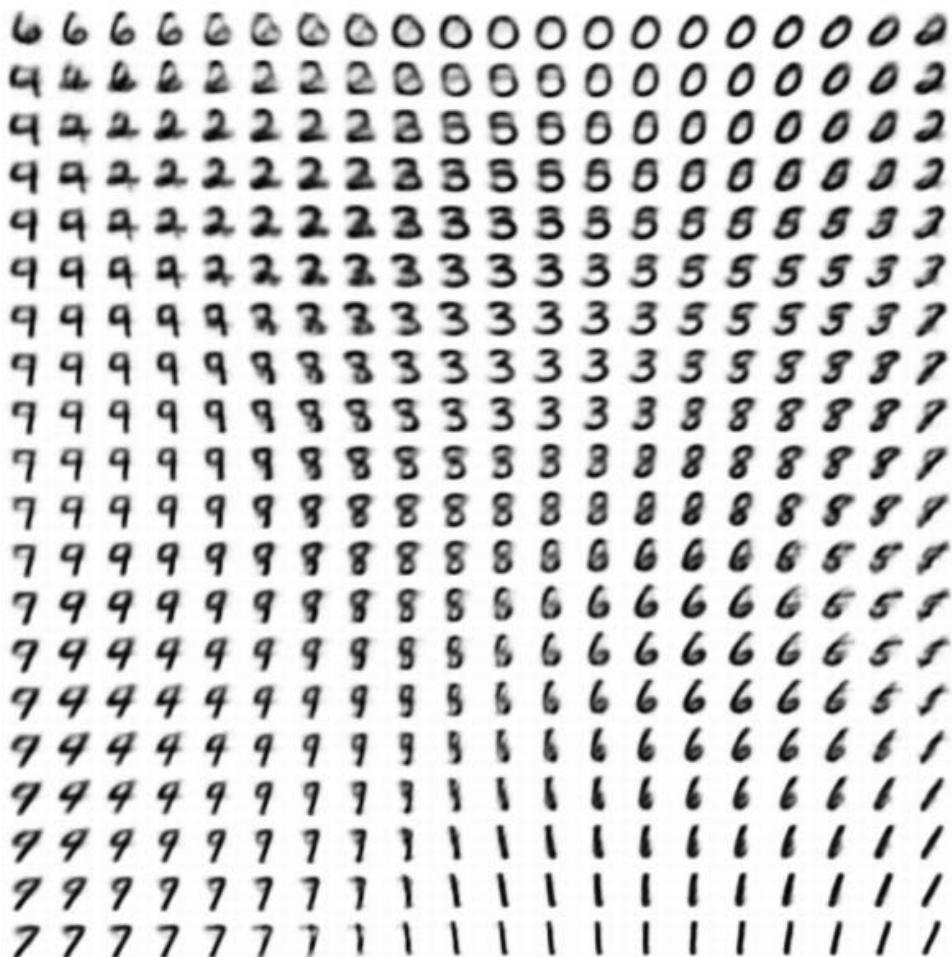


- 예를 들어 숫자의 잠재적인 의미를 생각하면
  - Digit(3인가 5인가)과 필체를 나타내는 잠재변수(각도, aspect ratio, 등)으로부터 숫자를 만들어낼 수 있다.

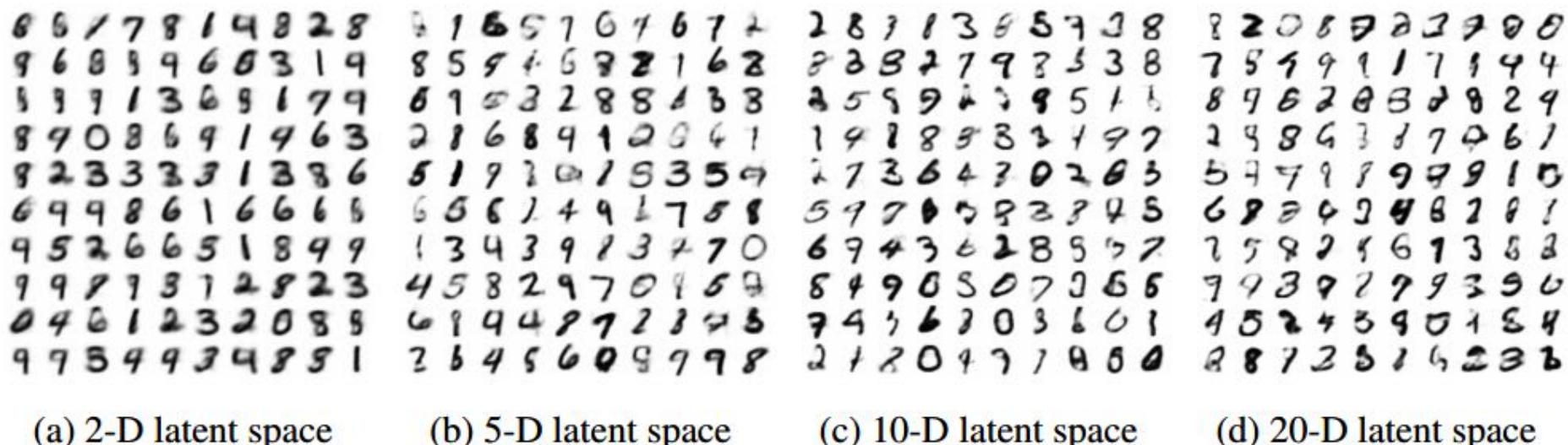




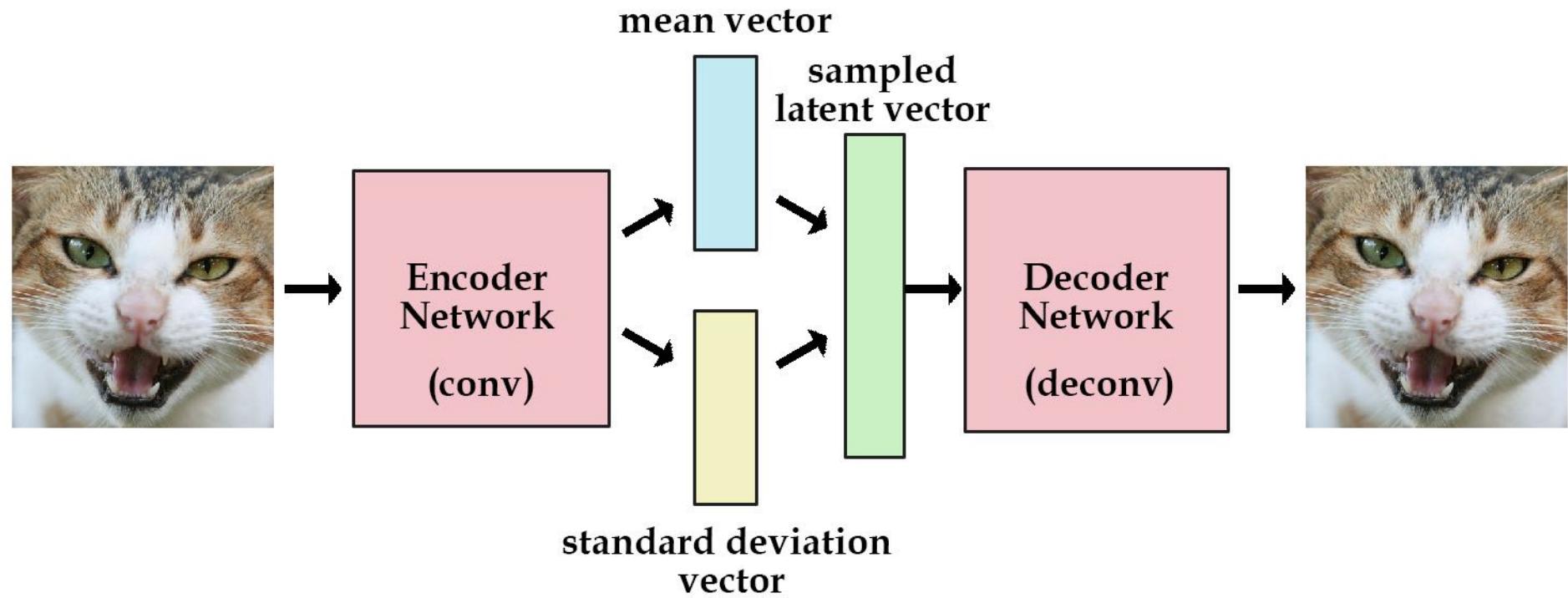
(a) Learned Frey Face manifold

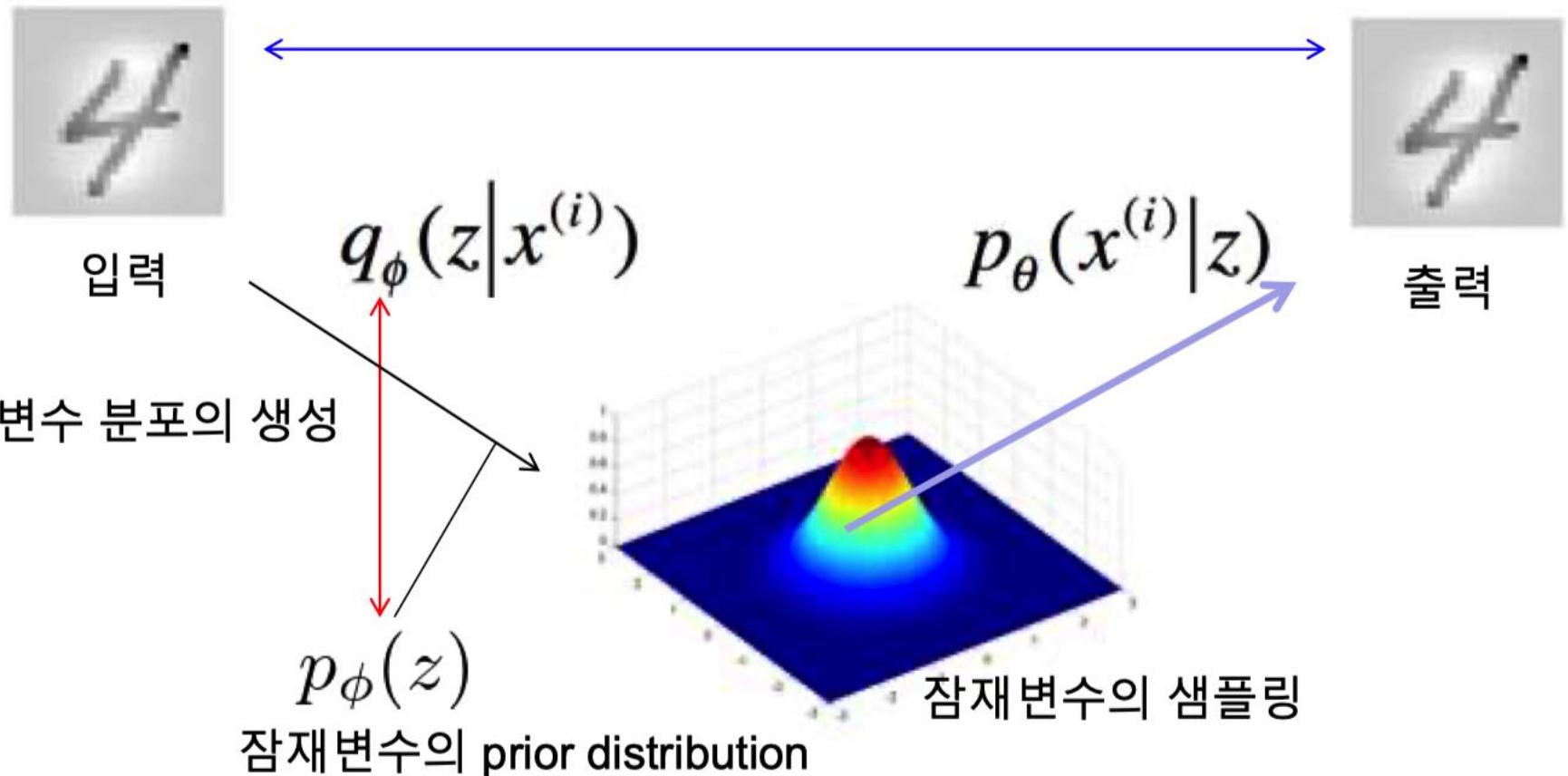


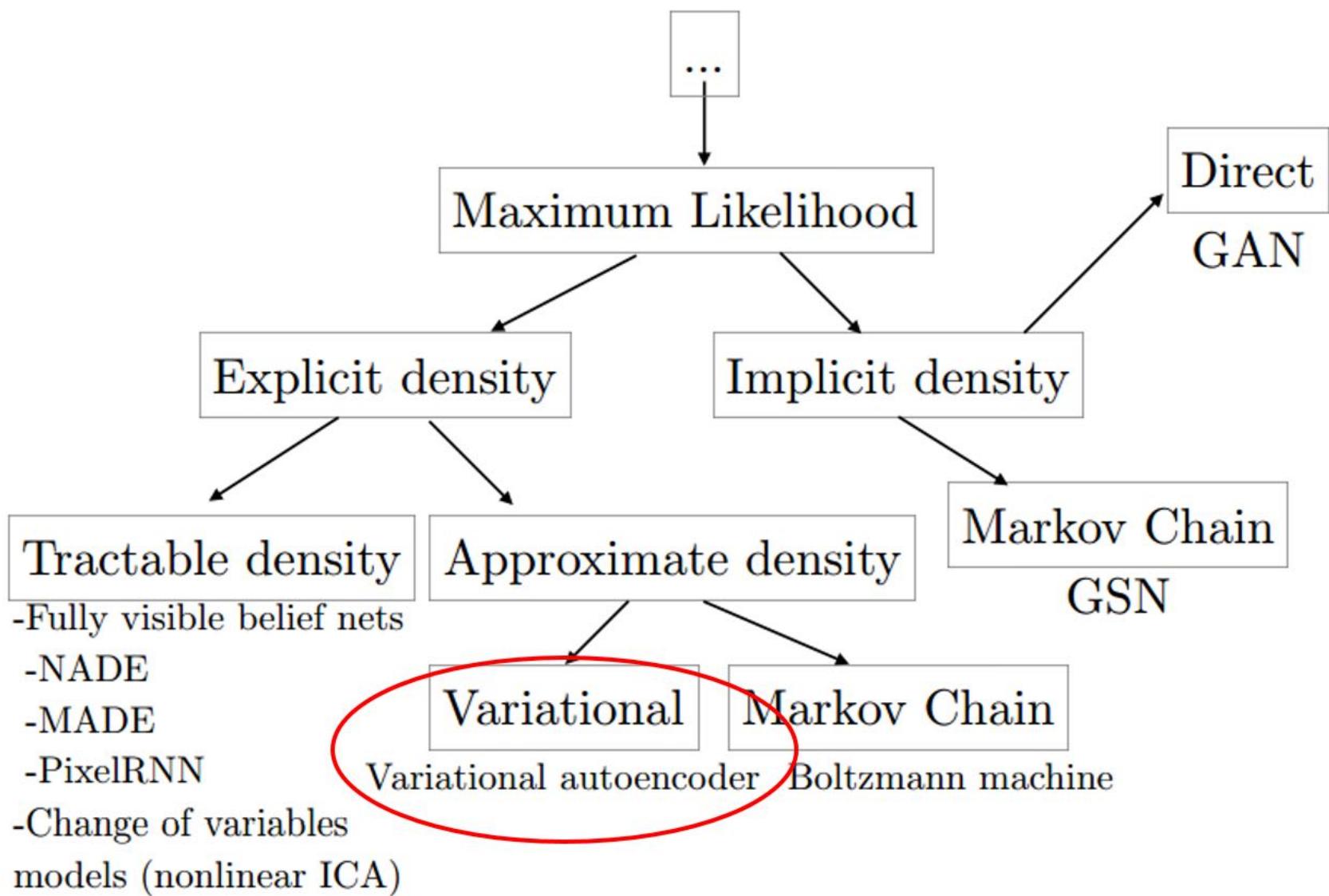
(b) Learned MNIST manifold



MNIST 데이터에 대해 각각 다른 latent space dimension으로 학습시키고  
random하게 값을 sample해본 결과  
dimension이 작은 model에서도 잘 나오는 것을 알 수 있음







# 간접적인(Implicit)

- 어떤 model에 대해 틀을 명확히 정의하는 대신 확률 분포를 알기 위해 sample을 뽑는 방법
  - Markov Chain
    - 알고있는 혹은 이미 얻은 sample  $x$ 와 transition operator  $q$ 가 주어졌을 때, sample  $x'$ 을 반복적으로 뽑다보면 결국에는 (일정 가정하에)  $x'$ 이  $p_{\text{model}}(x)$ 로부터 나온 sample로 수렴
  - GANs
    - 어떤 확률 모델(density)을 굳이 명확히 정의하지 않음
    - 모델(generator) 자체가 만드는 분포로부터 sample을 생성
      - MCMC와는 달리 별다른 input이 없이도 한 번에 sample을 생성할 수 있다는 것

# 직접적인(explicit)

- $p_{\text{model}}(x; \theta)$ 와 같이 model을 명확히 정의하여 이를 최대화하는 전략
- model을 정의했기 때문에 다루기가 비교적 편하고 어느 정도 모델의 움직임이 예측가능한 반면 아는 것 이상으로는 결과를 낼 수 없다는 한계가 있음
- 정의한 확률 모델이 계산이 가능(tractable)한지 혹은 불가능(intractable)한지에 따라
  - 모델이 직접 계산 가능한 경우는 사실 우리가 모든 변수를 다룰 수 있기 때문에 세밀한 조정이 가능한 동시에 사용할 수 있는 모델의 종류에 강력한 제약이 생깁니다. 이 것이 싫다면 좀 더 복잡한? 모델을 사용하면 되는데 그 대가로 계산이 불가능하여 density를 근사
  - 근사를 하는 방법에는 Monte Carlo approximations
    - 무작위로 많이 샘플을 뽑아서 분포를 유추해보자는 아이디어
    - 보통 모델에서 fair-sampling 하는 것이 쉽고 (계산량 측면에서) sample 간의 분산이 그리 높지 않은 경우 꽤나 잘 동작하는 방식

## 1. 데이터 구조의 가정과 모델의 근사가 필요

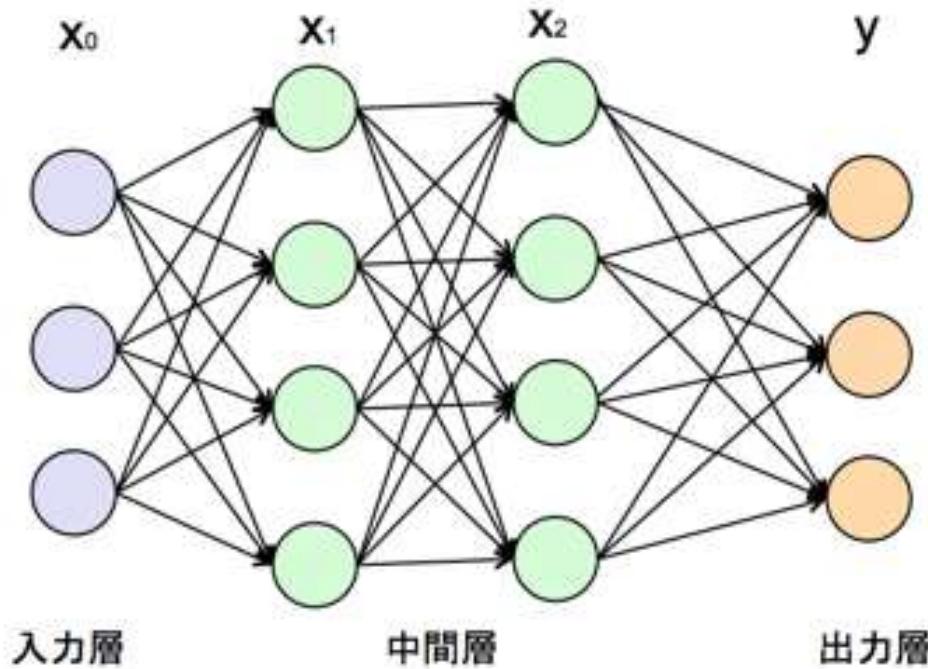
- 여기서 어떠한 분포의 설정이 필요
- 설정한 분포에 모델이 대응하여야 함

## 2. 시간이 소요되는 방법이 필요

- MCMC등과같이 복수의 샘플링이 필요

# 뉴럴넷의 이용

- 단순한 뉴럴넷의 예

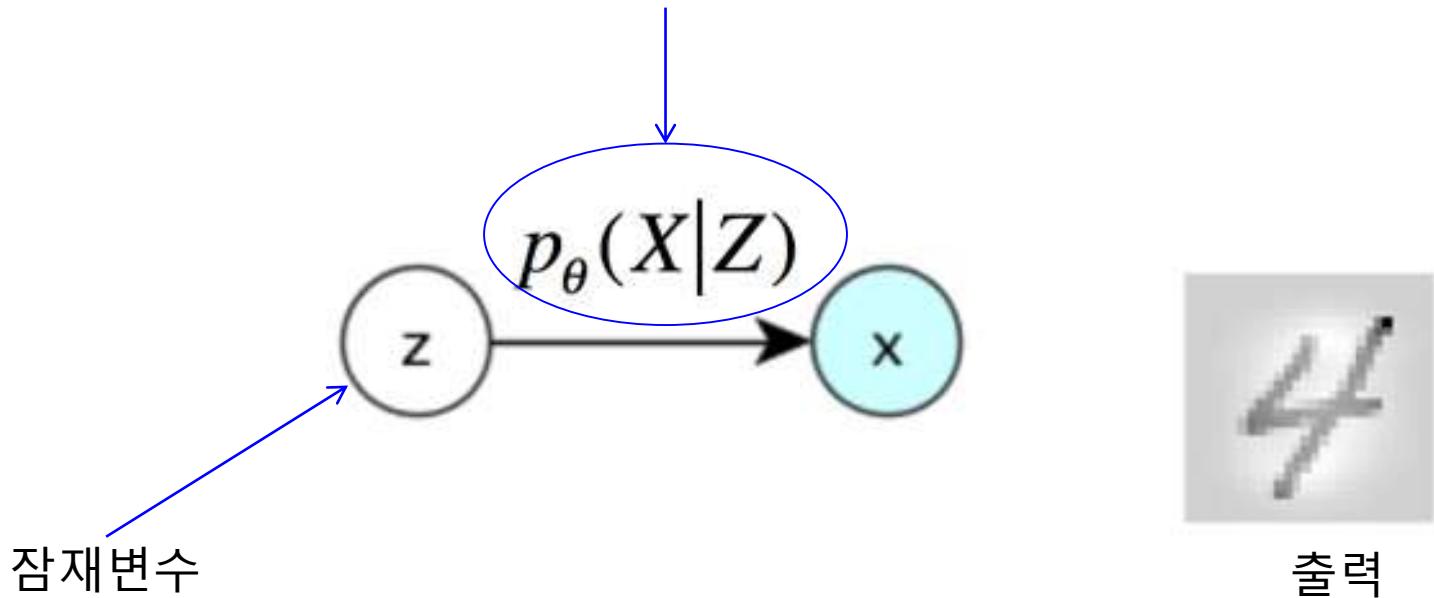


1.  $y = f_2(w_2x_2 + b_2) = f_2(f_2(w_2(f_1(w_1x_1 + b_1)) + b_2)) = \dots$   
→ Convolution형에서는 임의의 함수표현이 가능: 모델의 제약을 완화
2. SGD를 사용하면 1샘플씩 최적화가 가능

# 생성모델 최적화의 전제

- 원래

요것을 구하고 싶다(Z을 바탕으로 X가 생성되는 확률분포)

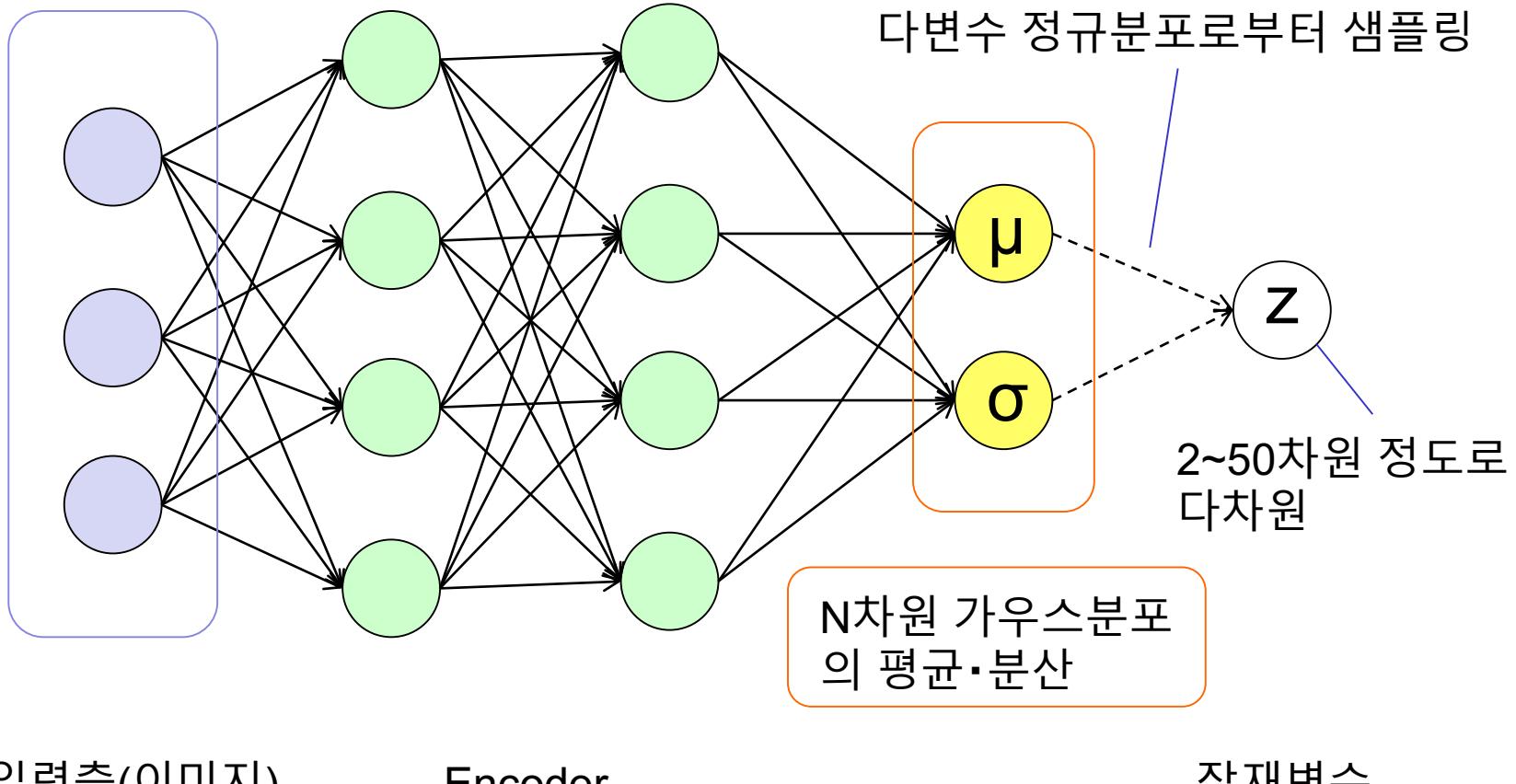


- 그러나, 이대로  $p_{\theta}$ 를 구하는 것은 곤란함
  - » 입력(잠재변수z)에 대응하는 답이 불명확함  
일반적으로 z은 저차원, X는 고차원임

# 입력에서 잠재변수 z으로의 분포를 가정

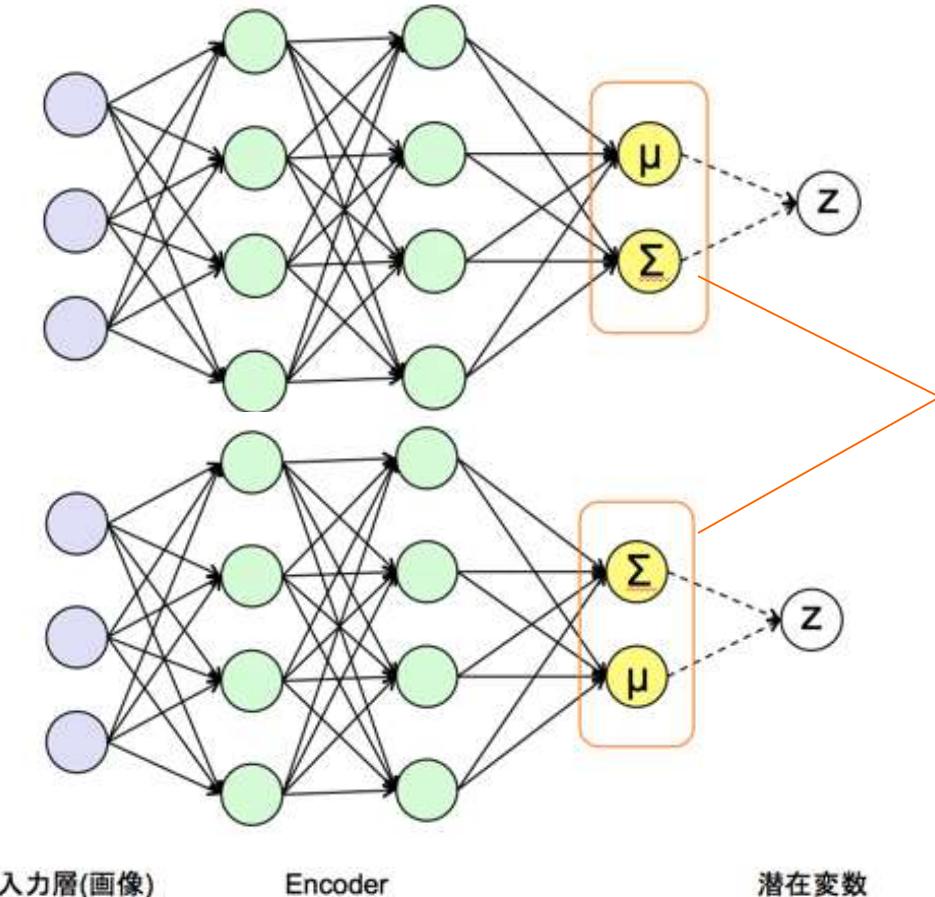
- Encoder
  - 잠재변수의 정규 분포를 가정

$$f(\mathbf{x}) = \frac{1}{(\sqrt{2\pi})^m \sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)$$



# 분포 parameter의 타당성

- $\mu$  와  $\sigma$ 의 결정에 타당성은 있나 ?
  - 어느 쪽이  $\mu \cdot \sigma$ 라도 좋다 :  $\mu$ 와  $\sigma$ 가 최적화되도록 NN을 최적화하면 됨



어떤 것이라도 괜찮다(미리  $\mu \cdot \sigma$  가 정의되어있지 않음)

나중에  $\mu \cdot \sigma$ 에 대응하도록 학습시킴

# GENERATIVE MODEL

Prior distribution  $p(z)$

Question: Is it enough to model  $p(z)$  with simple distribution like normal distribution?

Yes!!!

Recall that  $p(x|g_\theta(z)) = \mathcal{N}(x|g_\theta(z), \sigma^2 * I)$ . If  $g_\theta(z)$  is a multi-layer neural network, then we can imagine the network using its first few layers to map the normally distributed  $z$ 's to the latent values (like digit identity, stroke weight, angle, etc.) with exactly the right statistics. Then it can use later layers to map those latent values to a fully-rendered digit.

Generator가 여러 개의 레이어를 사용할 경우, 처음 몇 개의 레이어들을 통해 복잡할 수 있지만 딱 맞는 latent space로의 맵핑이 수행되고 나머지 레이어들을 통해 latent vector에 맞는 이미지를 생성할 수 있다.

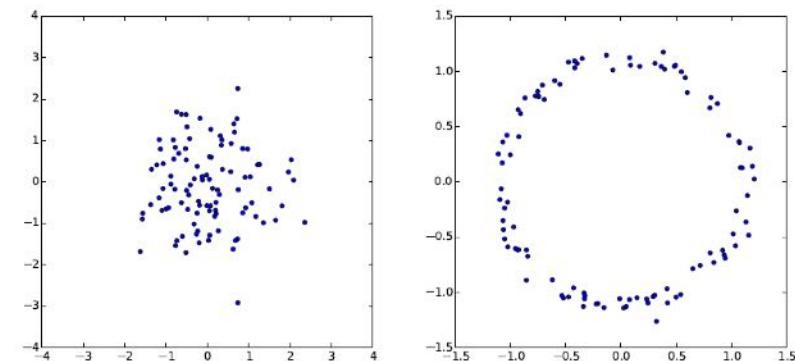
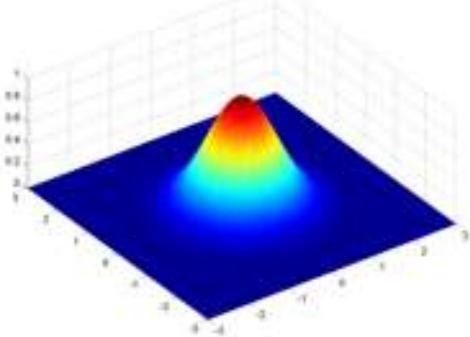


Figure 2: Given a random variable  $z$  with one distribution, we can create another random variable  $X = g(z)$  with a completely different distribution. Left: samples from a gaussian distribution. Right: those same samples mapped through the function  $g(z) = z/10 + z/\|z\|$  to form a ring. This is the strategy that VAEs use to create arbitrary distributions: the deterministic function  $g$  is learned from data.

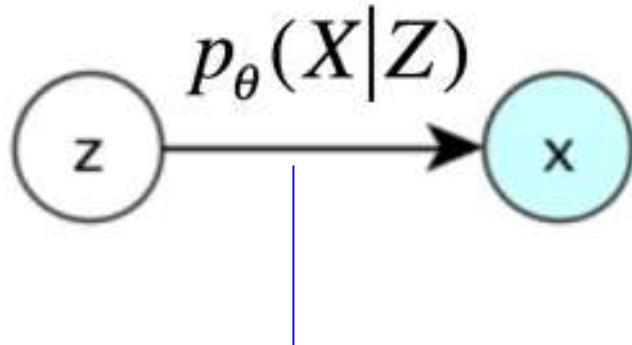
# 잠재변수의 가정

- 잠재변수는 다차원의 정규분포로 가정
    - 다루기 쉬움
    - 잠재변수로 문자의 필적과 형태를 가정 → 정규분포인가 ?
      - $z \sim p(z)$  :  $p$ 의 prior distribution으로 간단한 형태(다차원 표준 정규분포)

## 잠재변수의 분포



출력 이미지



잠재변수  $z$ 로 부터 이미지  $X$ 를 생성( $\theta$ 는 parameter)

# 최적화의 필요성

- 어떻게 최적화하여야 하나?
    - Maximum Likelihood Estimation : marginal likelihood  $\log(p_\theta(x))$ 의 최대화 되도록  
     $\Theta$ 를 정할 때 취할 수 있는  $x$ 의 Marginal probability가 가장 높도록
    - marginal likelihood  $\log(p_\theta(x))$ 는 다음과 같이 나눌 수 있음

$$\log p_\theta(x) = D_{KL}(q_\phi(z|x) || p_\theta(z|x)) + \underbrace{\mathcal{L}(\theta, \phi, x)}_{\substack{\uparrow \\ \text{variational lower limit : } \theta, \phi \text{의 함수}}} \quad \downarrow \quad \begin{array}{l} \text{일반적인 variational lower limit에 있어서 수식전개} \\ \text{수식전개} \end{array}$$

여기서 KL Divergence는

$$D_{KL}(q_\phi(z|x) || p_\theta(z)) \geq 0 \quad (\text{p}=\text{q} \text{시, 등호성립})$$

→ variational lower limit을 최대화하면 marginal likelihood도 커짐

# GENERATIVE MODEL

$$p_{\theta}(x|z)$$

**Question: Why don't we use maximum likelihood estimation directly?**

$$p(x) \approx \sum_i p(x|g_{\theta}(z_i))p(z_i)$$

If  $p(x|g_{\theta}(z)) = \mathcal{N}(x|g_{\theta}(z), \sigma^2 * I)$ , the negative log probability of X is proportional squared Euclidean distance between  $g_{\theta}(z)$  and  $x$ .

$x$  : Figure 3(a)

$z_{bad} \rightarrow g_{\theta}(z_{bad})$  : Figure 3(b)

$z_{bad} \rightarrow g_{\theta}(z_{good})$ : Figure 3(c) (identical to x but shifted down and to the right by half a pixel)

$$\|x - z_{bad}\|^2 < \|x - z_{good}\|^2 \rightarrow p(x|g_{\theta}(z_{bad})) > p(x|g_{\theta}(z_{good}))$$

Solution 1: we should set the  $\sigma$  hyperparameter of our Gaussian distribution such that this kind of erroneous digit does not contribute to  $p(X)$  → hard..

Solution 2: we would likely need to sample many thousands of digits from  $z_{good}$  → hard..

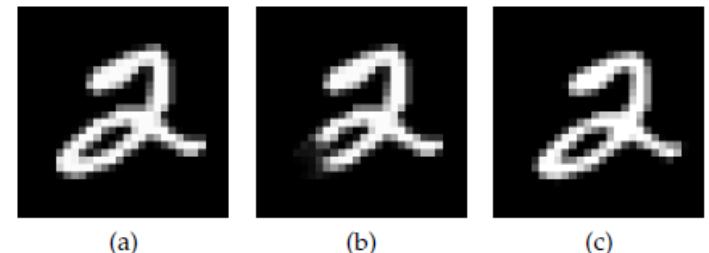


Figure 3: It's hard to measure the likelihood of images under a model using only sampling. Given an image X (a), the middle sample (b) is much closer in Euclidean distance than the one on the right (c). Because pixel distance is so different from perceptual distance, a sample needs to be extremely close in pixel distance to a datapoint X before it can be considered evidence that X is likely under the model.

생성기에 대한 확률모델을 가우시안으로 할 경우, MSE 관점에서 가까운 것이 더  $p(x)$ 에 기여하는 바가 크다. MSE가 더 작은 이미지가 의미적으로 더 가까운 경우가 아닌 이미지들이 많기 때문에 현실적으로 올바른 확률값을 구하기가 어렵다.

# Variational lower limit

- variational lower limit을 정리해보면
  - 아래와 같은 형태로 되며, 최적화 항이 도출됨

$$\begin{aligned}\mathcal{L}(\theta, \phi, x) &= \mathbb{E}_{q_\phi(z|x)} [\log p_\phi(x, z) - \log q_\phi(z|x)] \\ &= \mathbb{E}_{q_\phi(z|x)} [\log p_\phi(x|z) - p_\phi(z) - \log q_\phi(z|x)] \\ &= \text{정규화 항 : KL Divergence} \quad (\text{Regularization Parameter}) + \mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)]\end{aligned}$$

정규화 항 : KL Divergence  
(Regularization Parameter)

복원오차  
(Reconstruction Error)

이 두개의 합을 최대화하면 좋음

# 정규화 항 : KL Divergence

- KL Divergence의 계산

$$D_{KL}(\underline{q_\phi(z|x)} || \underline{p_\theta(z)})$$

$$\sim N(\mu, \sigma) \quad \sim N(0, I)$$

$$= D_{KL}(N(\mu, \Sigma) || N(0, I))$$

$$= -\frac{1}{2} \sum_{j=1}^J (1 + \log(\sigma_j^2) - \mu_j^2 + \sigma_j^2)$$

# 복원오차 : Reconstruction Error

- Reconstruction Error는 아래와 같이 근사화

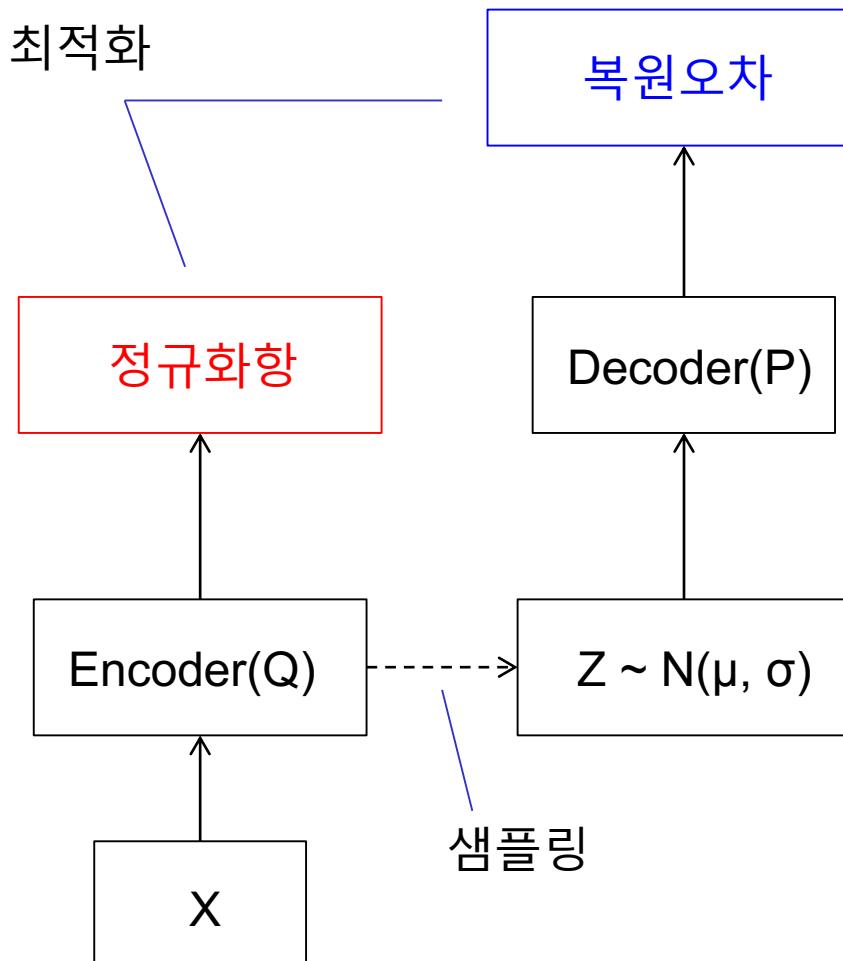
$$\mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)] \simeq \frac{1}{L} \sum_{l=1}^L \log p_\theta(x|z)$$

- 이미지 픽셀값을 0~1로 Normalize한 경우, Bernoulli분포로 가정하면  $\log p(x|z)$ 는 아래와 같이 나타낼 수 있음  
( $y$ 는 잠재변수  $z$ 를 Fully Connected Layer를 통과한 Final layer의 변수)

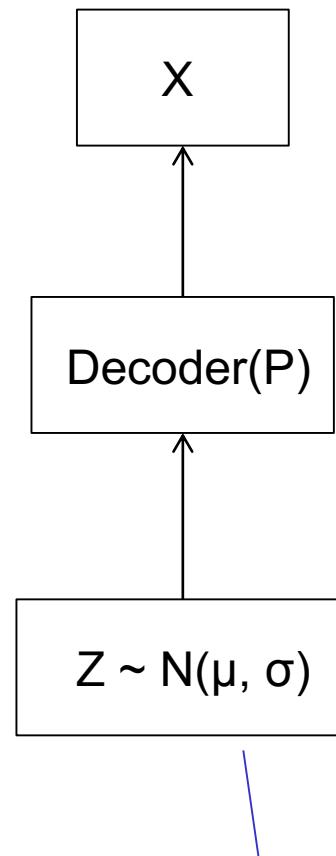
$$\log p(x|z) = \sum_{i=1}^D x_i \log y_i + (1 - x_i) \cdot \log(1 - y_i)$$

# VAE 전체 block diagram

학습 단계

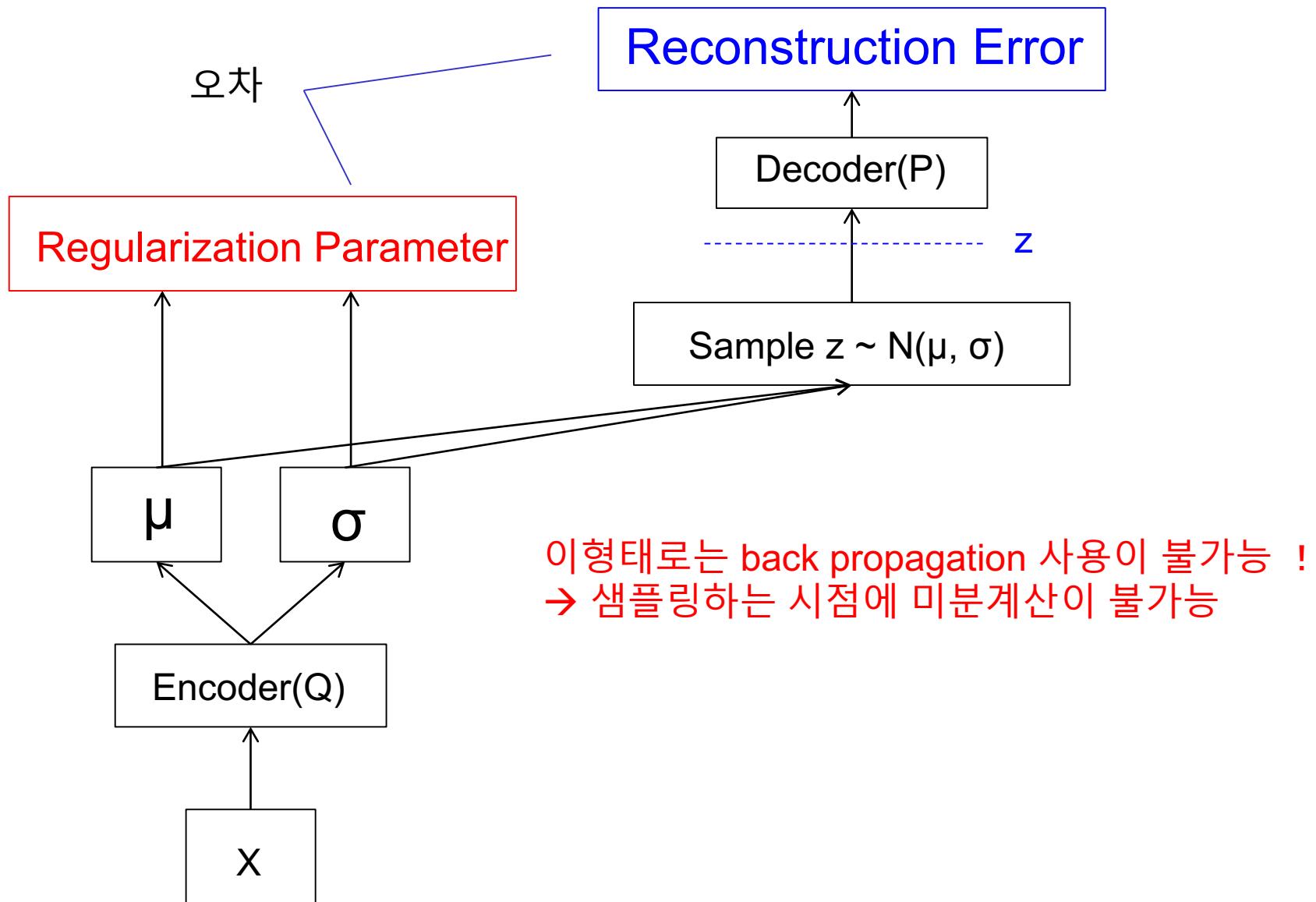


활용 단계

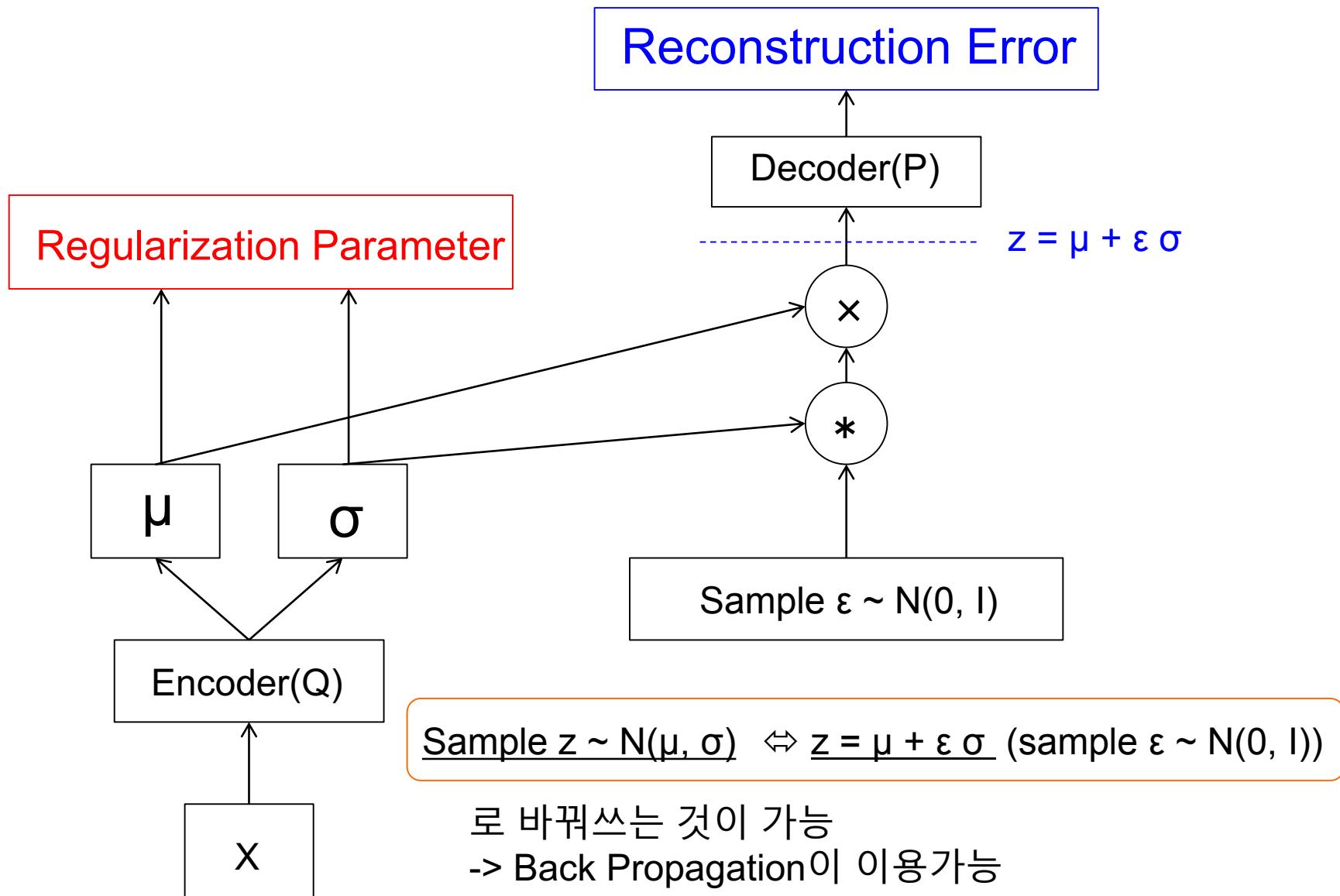


$Z$ 를 입력으로서 준다

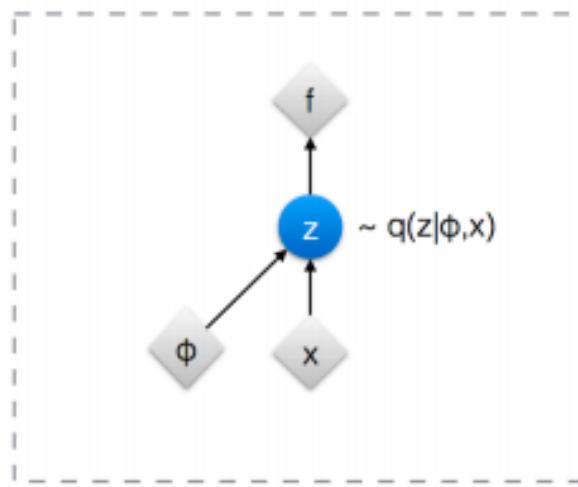
# 학습 단계의 보다 상세한 구조



# Reparametrization Trick



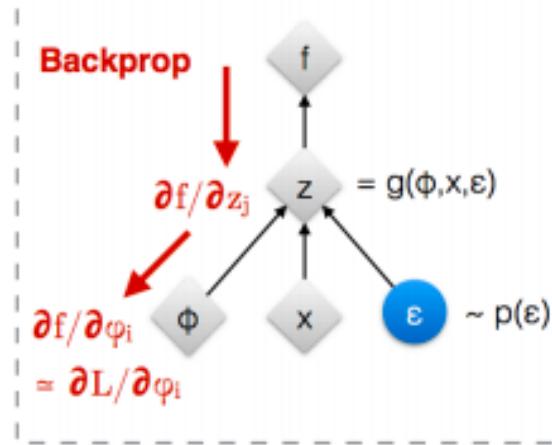
## Original form



◆ : Deterministic node

● : Random node

## Reparameterised form



[Kingma, 2013]

[Bengio, 2013]

[Kingma and Welling 2014]

[Rezende et al 2014]

# Z의 변환에 대하여

- 일차원 경우의 간단한 증명

Sample  $z \sim N(\mu, \sigma)$   $\Leftrightarrow$   $z = \mu + \epsilon \sigma$  (sample  $\epsilon \sim N(0, 1)$ )

$\epsilon$  는 표준 정규분포이므로 확률밀도함수는  $f(\epsilon) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\epsilon^2}{2}\right)$

$z = \mu + \epsilon \cdot \sigma \Leftrightarrow \epsilon = \frac{z - \mu}{\sigma}$ 로 변환 가능하므로 대입하면

$$f(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(z - \mu)^2}{2\sigma^2}\right)$$

정규분포  $z \sim (N(\mu, \sigma))$ 로부터 샘플링과 동일

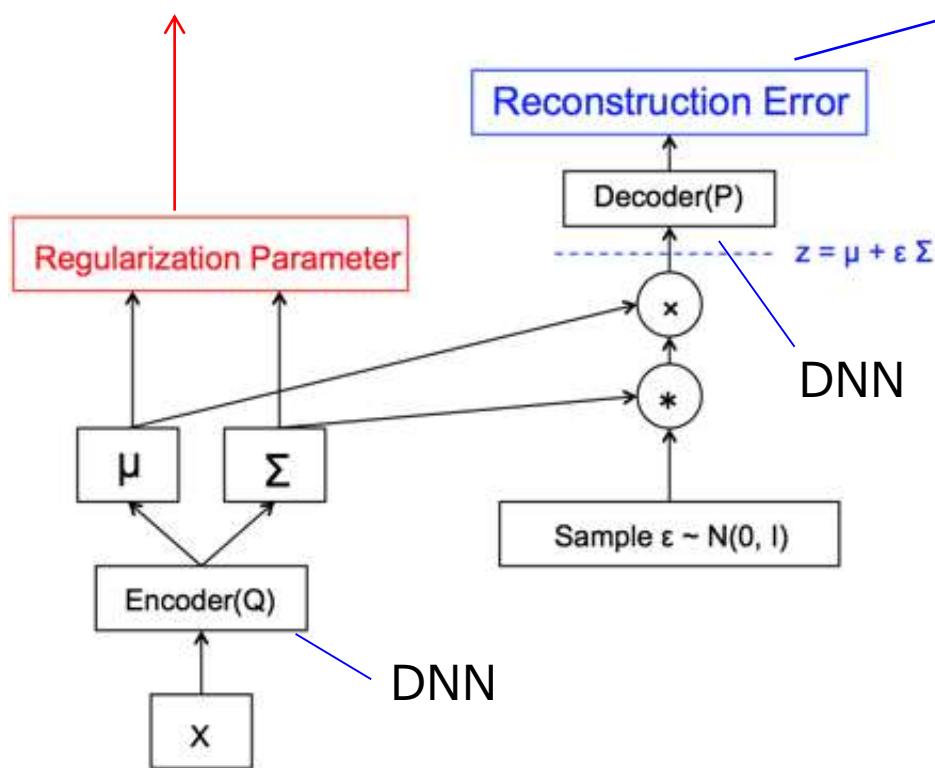
차수가 2차 이상인 경우도 동일

# VAE의 최적화 정리

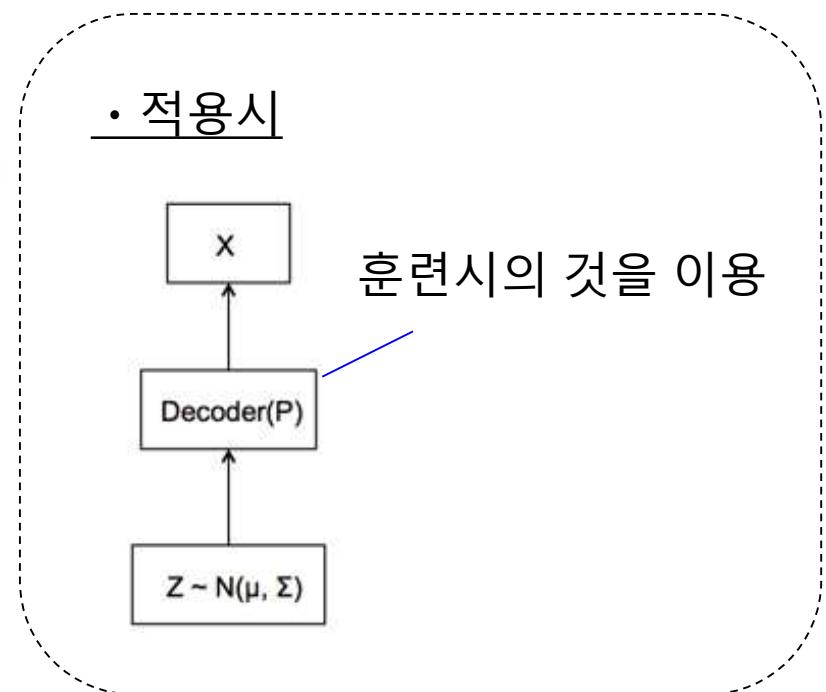
## • 훈련시

$$\mathcal{L}(\theta, \phi, x)$$

$$= -\frac{1}{2} \sum_{j=1}^J (1 + \log(\sigma_j^2) - \mu_j^2 + \sigma_j^2) + \mathbb{E} \left[ \sum_{i=1}^D (x_i \log y_i + (1 - x_i) \cdot \log(1 - y_i)) \right]$$



## • 적용시



훈련시의 것을 이용

## Variational Bayes 방식

모델인 posterior  $q(z|x)$ 를 세워 intractability 문제를 해결하였으나 lower bound를 좀 더 maximize 해서 gap을 더 줄이기 위해서는 likelihood  $p(x|z)$ 를 사람이 좀더 똑똑한 방식으로 열심히 디자인해서 새로 갈아껴보고 하는 방식으로 연구들이 진행  
따라서 적절한 모델을 디자인하기 위해 domain knowledge가 매우 중요

## VAE는 DNN을 optimization 문제로

1. variational inference의 아이디어를 이용해서 기존 문제를 optimization으로 바꿔주고,
2. 이렇게 바뀐 문제를 위와 같은 방식 대신  $q(z|x)$ 와  $p(x|z)$ 가 각각 encoder와 decoder인 Auto-Encoder로 모델링하여
3. 이를 gradient descent를 사용하여 둘 다 한 방에 update하였다는 점에서 당시로서는 매우 획기적인 발상의 전환

## VAE의 장단점

- VAE는 GAN에 비해 학습이 안정적인 편
  - 손실함수에서 확인할 수 있듯 *reconstruction error*과 같이 평가 기준이 명확하기 때문
- 데이터뿐 아니라 데이터에 내재한 잠재변수  $zz$ 도 함께 학습할 수 있다는 장점(*feature learning*).
  - 출력이 선명하지 않고 평균값 형태로 표시
    - MLE를 사용한 것으로 설명하는 쪽도 있지만 꼭 이렇다할 명확한 이유를 제대로 밝힌 연구는 없음
    - GAN이 보다 Sharp한 이미지를 생성할 수 있는지에 대한 논의도 여전히 진행 중
  - *reparameterization trick*이 모든 경우에 적용되지 않음