

$q_\theta(Z|X)$
Variational Inference

VAE 알아보기

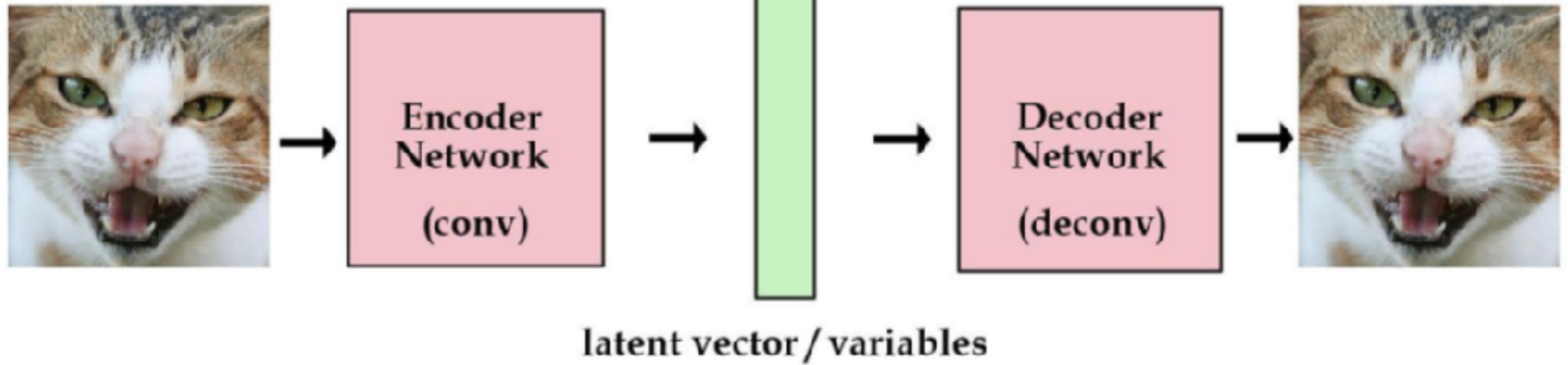
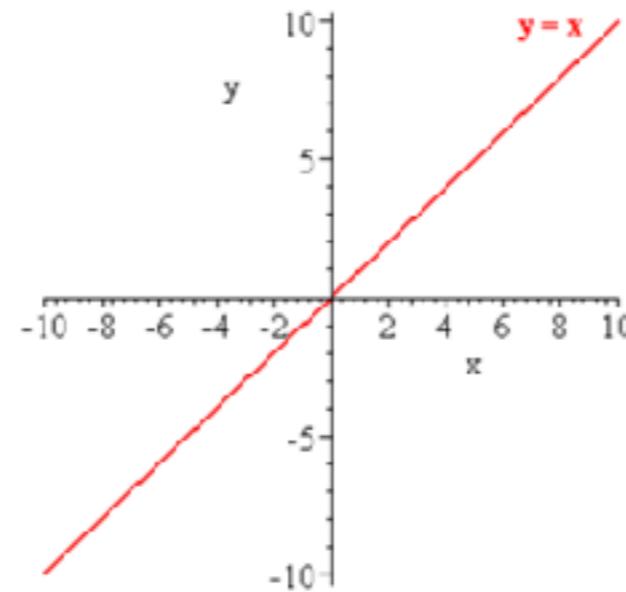
KL-divergence ..

정보이론 ..

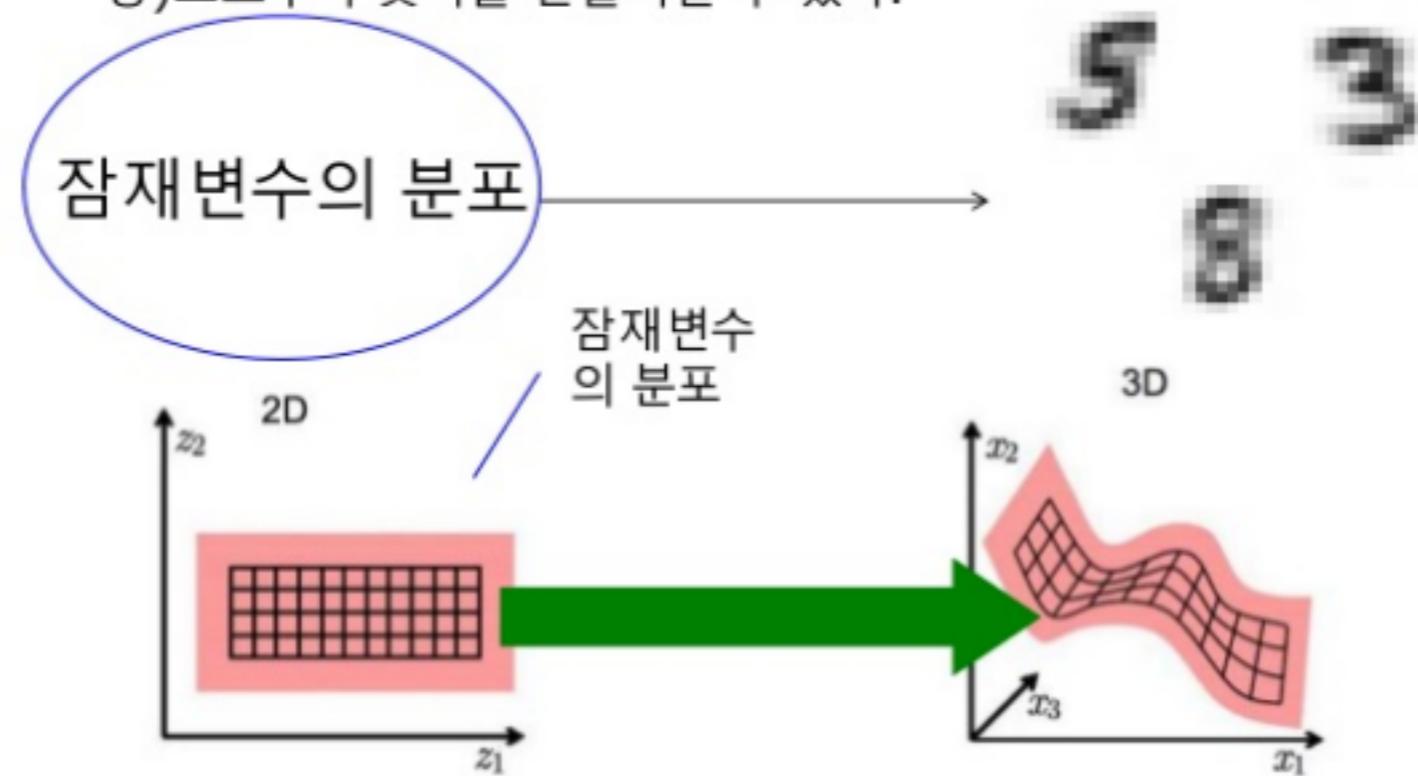
Manifold ..

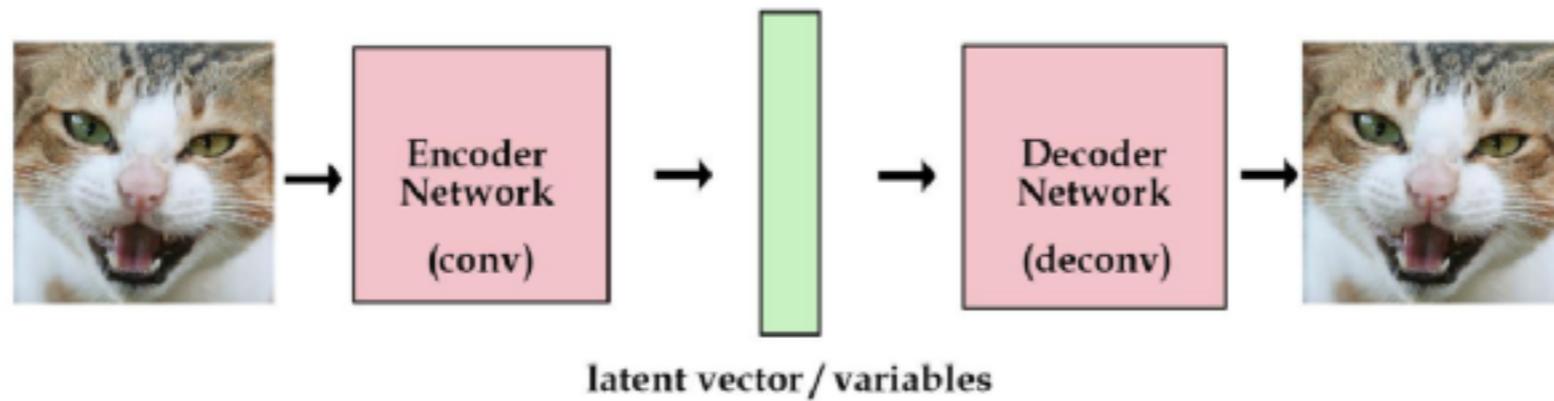
Variational inference 몰라요.

Inference 알고리즘 몰라요.

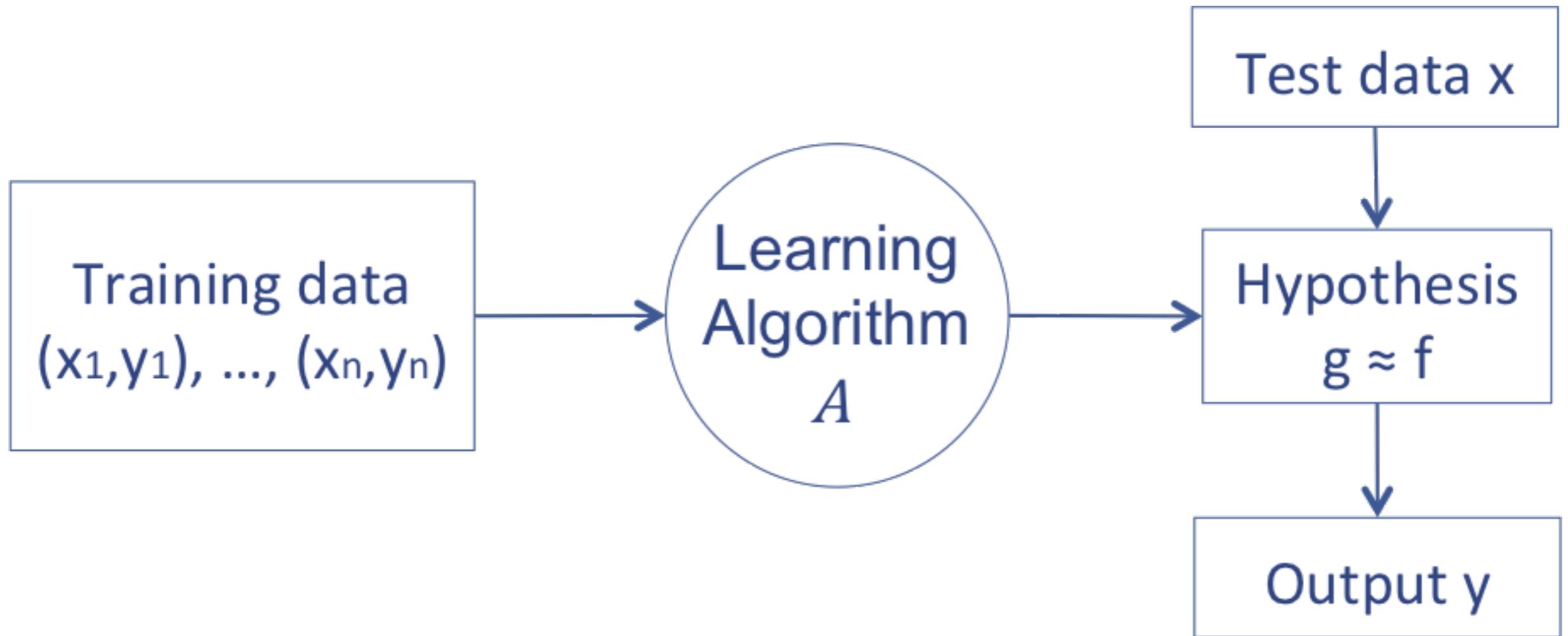


- 예를 들어 숫자의 잠재적인 의미를 생각하면
 - Digit(3인가 5인가)과 필체를 나타내는 잠재변수(각도, aspect ratio, 등)으로부터 숫자를 만들어낼 수 있다.



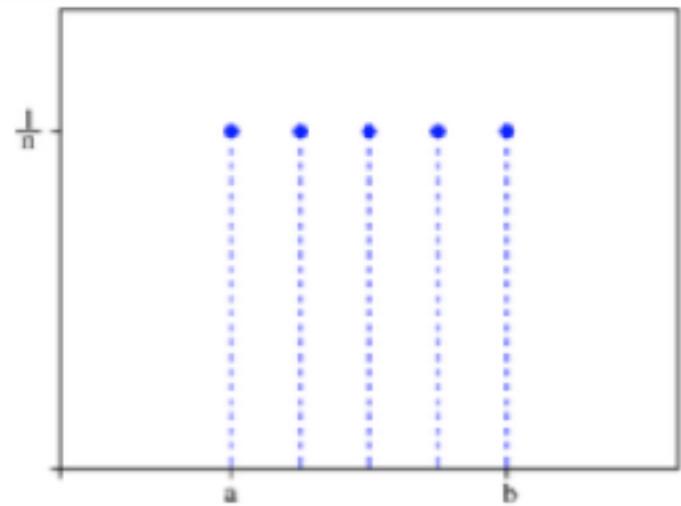


However, we're trying to build a generative model here, not just a fuzzy data structure that can "memorize" images. We can't generate anything yet, since we don't know how to create latent vectors other than encoding them from images.



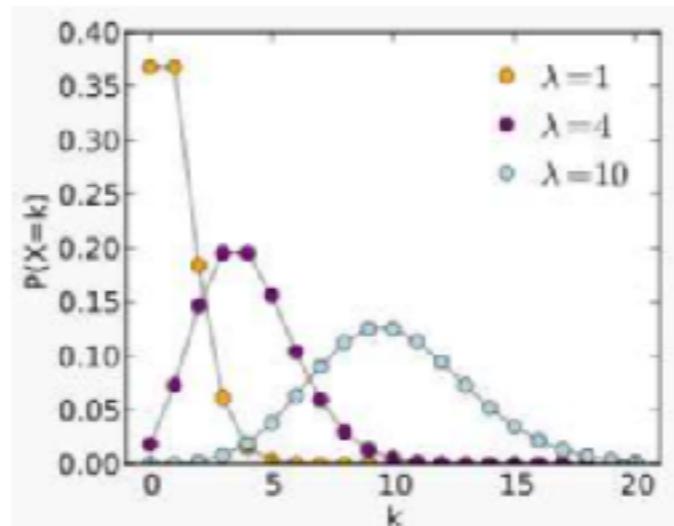
이전 글에서 나는 머신러닝이 주어진 데이터를 가장 잘 설명하는 ‘함수’를 찾는 알고리즘을 디자인하는 것이라 설명했다. 그러나 머신러닝은 확률의 관점에서도 설명이 가능하다. 머신러닝을 [probability density](#)를 찾는 과정으로 생각하는 것이다. 즉, 함수를 가정하는 것이 아니라 확률 분포를 가정하고, 적절한 확률 분포의 parameter를 유추하는 과정으로 생각하는 것이다.

주어진 데이터가 gaussian distribution으로 drawn되었다고 가정하고, 데이터와 현상을 가장 잘 설명하는 mean과 covariance를 찾는 과정과 비슷한 것이라고 생각하면 된다. 즉, 앞에서 설명한 방식은 function parameter를 찾는 방식이라면, 이제는 probability density function parameter를 찾는 것으로 바꿔는 것이다.



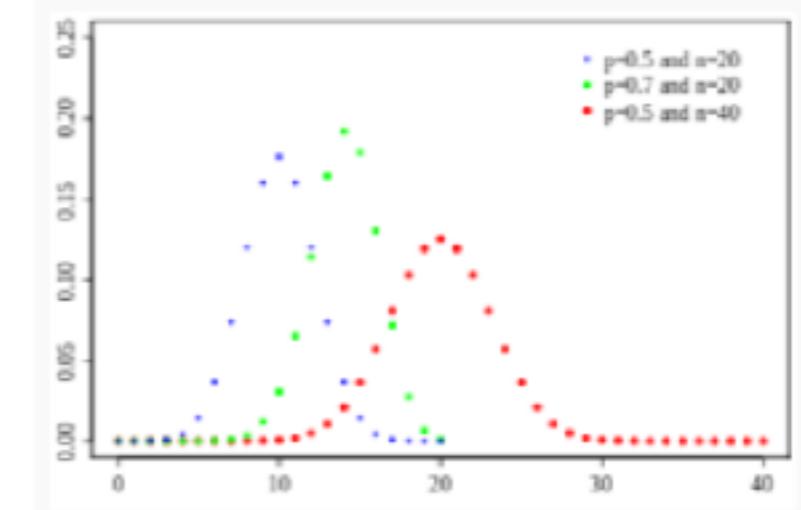
이산 균등분포

$$\frac{1}{n}$$



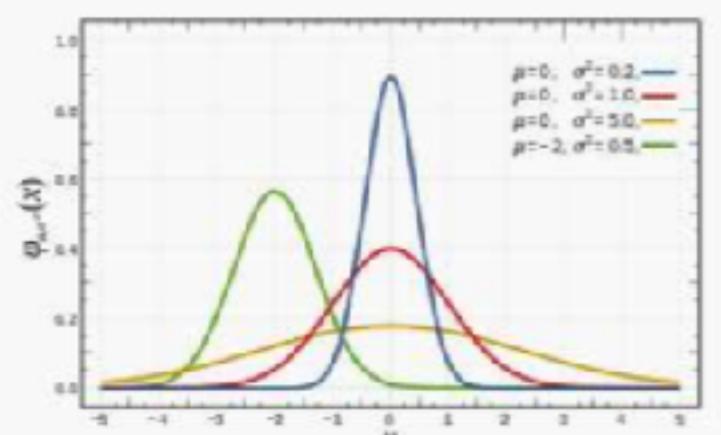
포아상분포

$$f(n; \lambda) = \frac{\lambda^n e^{-\lambda}}{n!},$$



이항분포

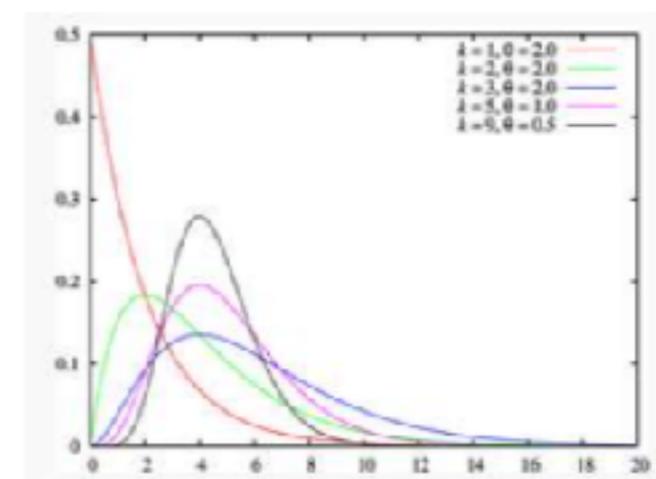
$$\Pr(K=k) = f(k; n, p) = \binom{n}{k} p^k (1-p)^{n-k}$$



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



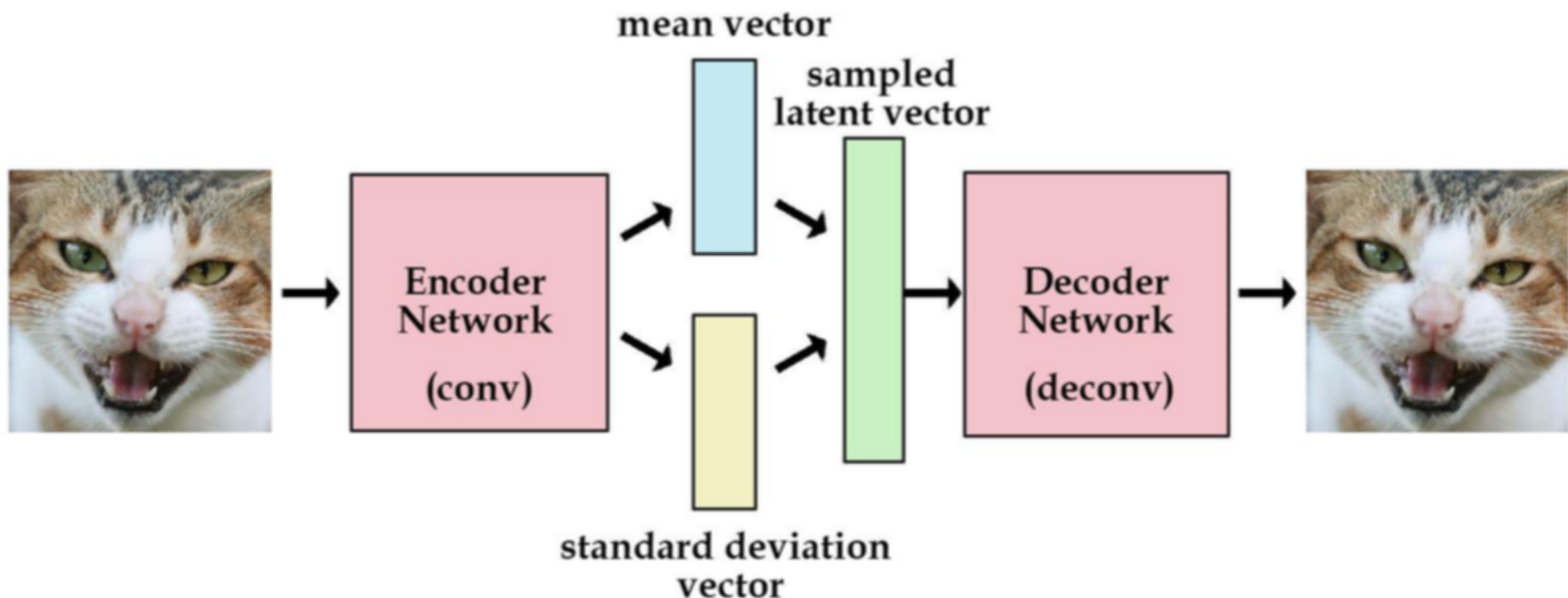
연속 균등분포

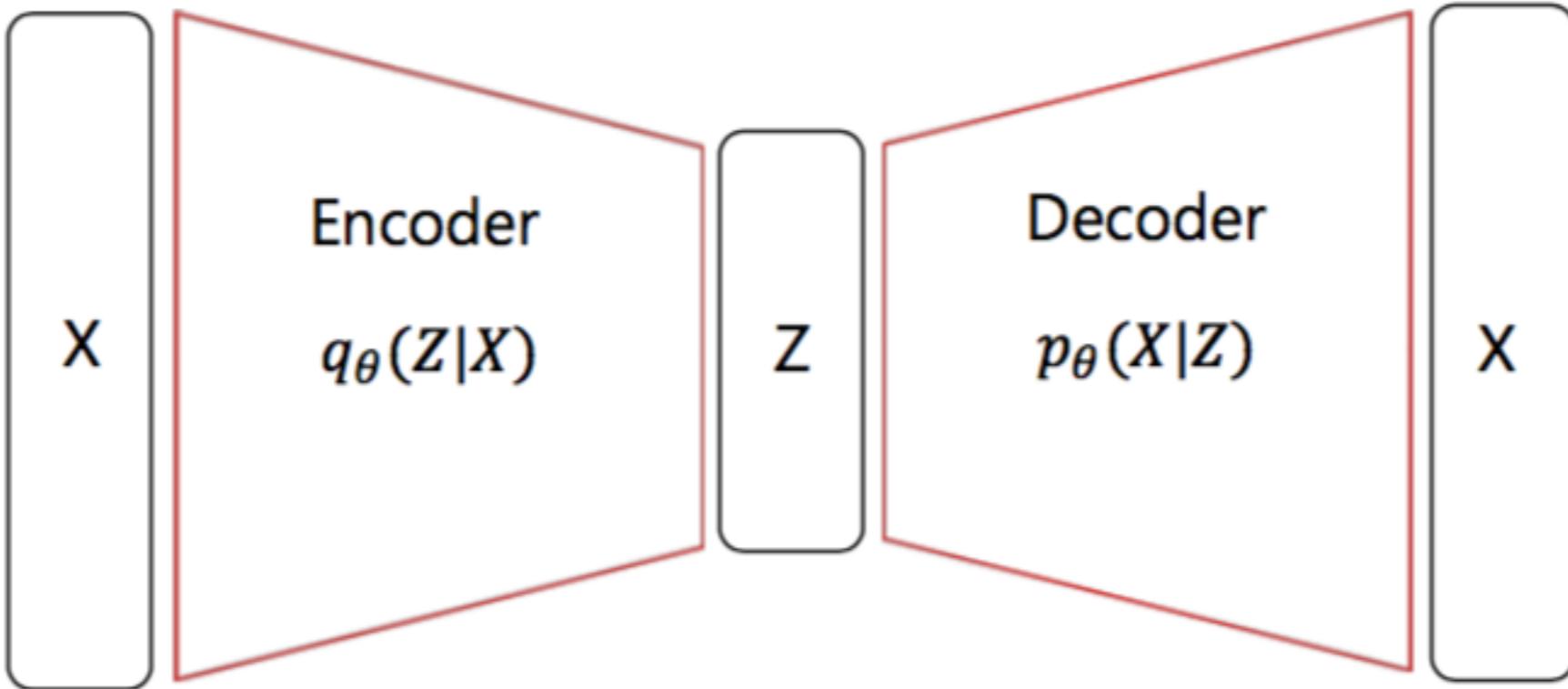


감마분포

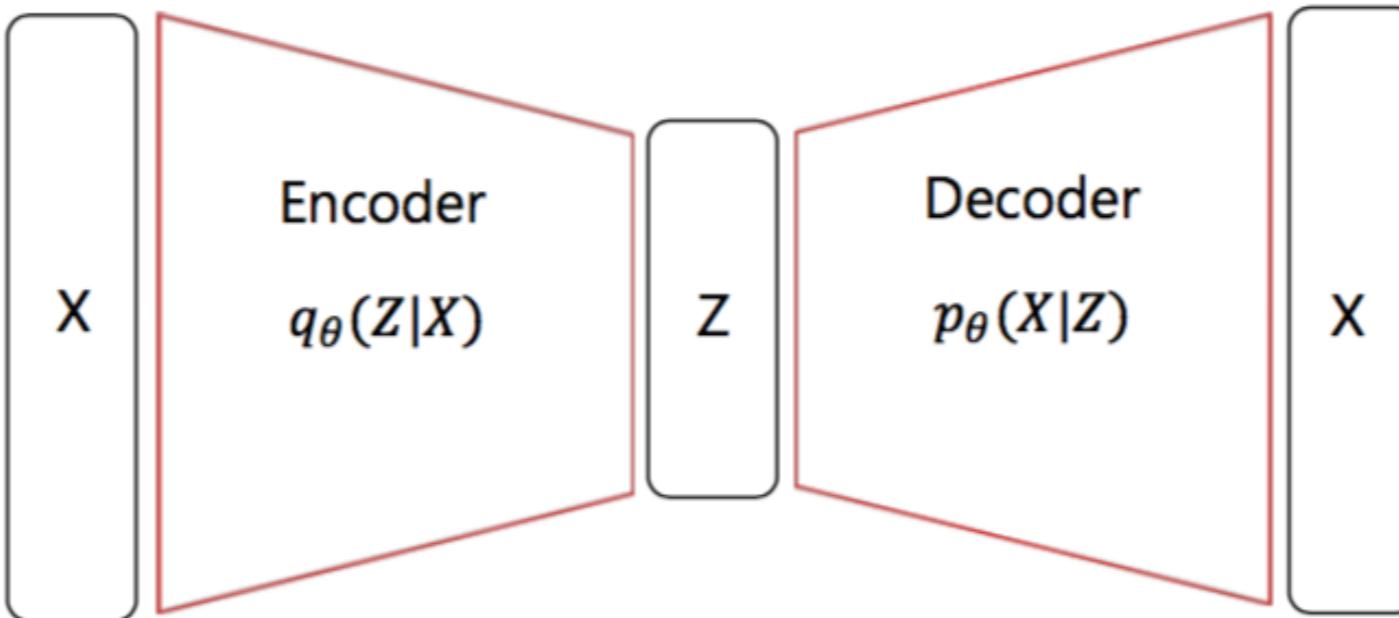
$$f(x; k, \theta) = x^{k-1} \frac{e^{-x/\theta}}{\theta^k \Gamma(k)} \text{ for } x > 0$$

There's a simple solution here. We add a constraint on the encoding network, that forces it to generate latent vectors that roughly follow a unit gaussian distribution. It is this constraint that separates a variational autoencoder from a standard one.



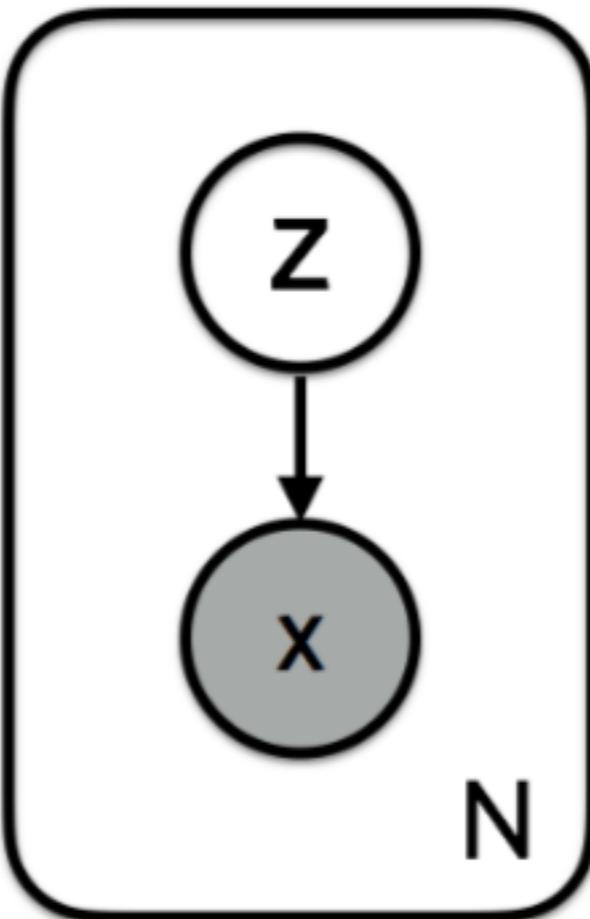


Encoder는 뉴럴넷이다. 인풋은 datapoint x 이고, 아웃풋은 잠재된 차원 (hidden representation) z 이며, weight들과 biase θ 를 가지고 있다. 구체적으로 설명하기 위해, x 를 28×28 사이즈의 손으로 쓴 숫자 사진이라고 생각하자. Encoder는 784 차원 데이터를 훨씬 낮은 차원의 잠재된 차원 z 로 암호화(encode)한다. 이는 흔히 병목 지역이라 불리는데, encoder가 데이터를 낮은 차원으로 효율적으로 압축하는 것을 배워야 하기 때문이다. Encoder를 $q_{\theta}(z|x)$ 로 표현하자. 낮은 차원 (z)이 확률론적임을 확인하자: encoder는 Gaussian probability density인 $q_{\theta}(z|x)$ 에 파라미터를 아웃풋으로 내보낸다. 우리는 이 확률 분포에서 샘플을 뽑음으로써 z 에 잡음이 포함된 값을 얻을 수 있다.



Decoder는 또 다른 뉴럴넷이다. 인풋은 z 이고 아웃풋은 데이터 확률분포의 파라미터 값들이며 weight과 biase ϕ 를 갖는다. Decoder는 $p_\phi(x|z)$ 로 표현된다. 손으로 쓴 숫자 사진을 사용한다고 할 때, 사진은 흑백이고 각 픽셀이 0이나 1이라 하자. 그러면 픽셀 하나의 확률분포는 Bernoulli distribution으로 표현할 수 있다. Decoder는 숨겨진 차원 z 을 input으로 받아 784개의 각각 이미지 픽셀에 대한 Bernoulli 파라미터들을 output으로 내보낸다. 실수로 된 z 를 784개의 0과 1 사이의 실수로 해독(decode)하는 것이다. 낮은 차원에서 높은 차원으로 가기 때문에 정 보의 손실이 발생한다. 얼마나 손실이 발생하는가? 우리는 이 것을 단위가 자연수인 reconstruction log-likelihood $\log p_\phi(x|z)$ 를 사용하여 측정한다. 이 측정값이 decoder가 숨겨진 차원 z 에서 인풋 x 를 얼마나 효율적으로 재생성 했는지를 알려준다.

BAYES



VAE의 그래프 모델. 숨겨진 변수 z 는 표준정규분포이며 $data$ 는 $P(X|Z)$ 에서 추출된다. 그림자진 노드 X 는 관찰된 $data$ 를 의미한다. 손글씨 숫자 이미지에 대해서 이 $data$ likelihood는 *Bernoulli distribution*을 따른다.

확률 모델의 관점에서 VAE는 어떤 특정한 확률 모델을 갖는 $data x$ 와 숨겨진 변수(latent variable) z 를 갖는다. 우리는 이 모델의 joint probability를 $p(x, z) = p(x|z)p(z)$ 로 표현할 수 있다. 생성 과정은 다음과 같다.

이 모델이 우리가 VAE 를 확률 모델의 관점에서 생각할 때 가장 중요한 부분이다. 숨겨진 변수는 prior $p(z)$ 로부터 얻어진다. 데이터 x 는 숨겨진 변수 z 를 조건부로 한 likelihood $p(x|z)$ 를 갖는다. 모델은 data 와 숨겨진 변수에 대한 joint probability $p(x, z)$ 를 정의 한다. 우리는 이를 likelihood 와 prior 로 분해 할 수 있다: $p(x, z) = p(x|z)p(z)$ 흑백 숫자 이미지의 경우, likelihood 는 Bernoulli distribution 을 따른다.

$p(x|z)$

Bernoulli

Parameters	$0 < p < 1, p \in \mathbb{R}$
Support	$k \in \{0, 1\}$
pmf	$\begin{cases} q = (1 - p) & \text{for } k = 0 \\ p & \text{for } k = 1 \end{cases}$
CDF	$\begin{cases} 0 & \text{for } k < 0 \\ 1 - p & \text{for } 0 \leq k < 1 \\ 1 & \text{for } k \geq 1 \end{cases}$
Mean	p
Median	$\begin{cases} 0 & \text{if } q > p \\ 0.5 & \text{if } q = p \\ 1 & \text{if } q < p \end{cases}$
Mode	$\begin{cases} 0 & \text{if } q > p \\ 0, 1 & \text{if } q = p \\ 1 & \text{if } q < p \end{cases}$
Variance	$p(1 - p) (= pq)$

모수는 p

https://en.wikipedia.org/wiki/Bernoulli_distribution

<http://nolsigan.com/blog/what-is-variational-autoencoder/>

이제 우리는 이 모델에서의 inference에 대해 생각해 볼 수 있다. 목표는 관찰된 데이터로부터 숨겨진 변수를 잘 추론(infer)하는 것이다. 곧, posterior $p(z|x)$ 를 계산하는 것과 일치한다. Bayes에 따르면:

$$p(x, z) = p(x|z)p(z)$$

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

Likelihood **Prior**
Posterior **Evidence**

각각은 *observation(likelihood)*, 현상에 대한 사전정보 (*prior*), 주어진 데이터에 대한 현상의 확률 (*posterior*)을 의미한다.

$$\frac{\text{Likelihood}}{\text{Prior}} = \frac{p(x|z)p(z)}{p(x)}.$$

Posterior **Evidence**

$$p(x, z) = p(x|z)p(z)$$

모든 가능한 Z 에 대해,
 $p(x,z)$ 와 $p(z)$ 를 구해야 함

분모 $p(x)$ 를 생각해보자. 우리는 이를 evidence 라 부르고 숨겨진 변수를 제거하여 얻을 수 있다: $p(x) = \int p(x, z)p(z)dz$. 운이 없게도, 이 적분을 계산하기 위해선 모든 숨겨진 변수의 가능성을 고려해야 하기 때문에 지수 시간이 걸린다. 따라서 우리는 이 posterior distribution 을 구해야 한다.

Variational inference 가 $q_\lambda(z|x)$ 계열의 분포를 근사할 수 있다.

Variational parameter λ 는 분포 계열을 구별할 수 있게 한다. 예를 들어, q 가 Gaussian 이라면, 각 datapoint x_i 에 대한 숨겨진 변수의 평균과 분산이 된다. $\lambda_{x_i} = (\mu_{x_i}, \sigma_{x_i}^2)$

우리의 **variational posterior** $q(z|x)$ 가
진짜 **posterior** $p(z|x)$ 를 얼마나 잘 근사하는지 어떻게 알 수 있을까?

확률과 정보량

확률에 대한 정보량

$$h(x) = -\log\{p(x)\}$$

- 주사위 눈금이 1이 나올 확률에 대한 정보량

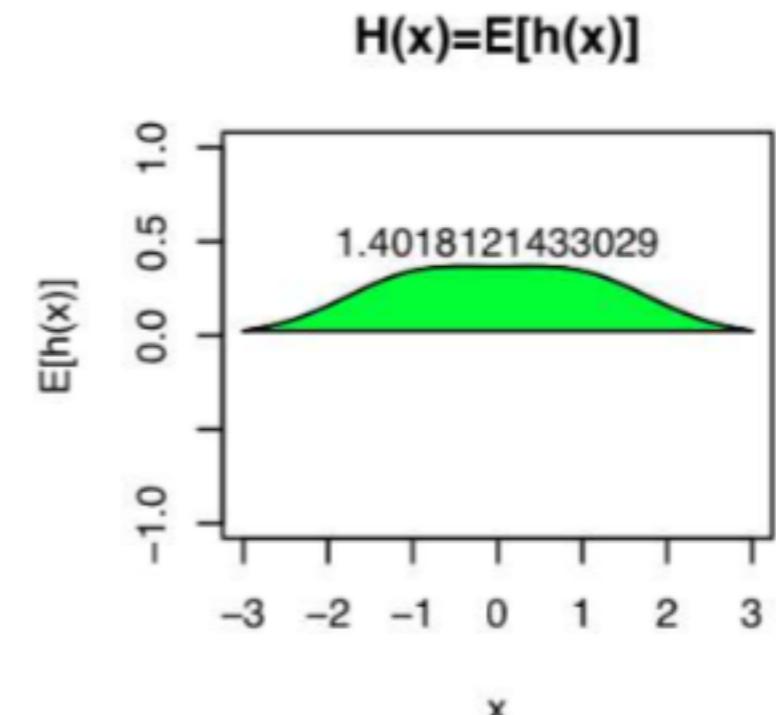
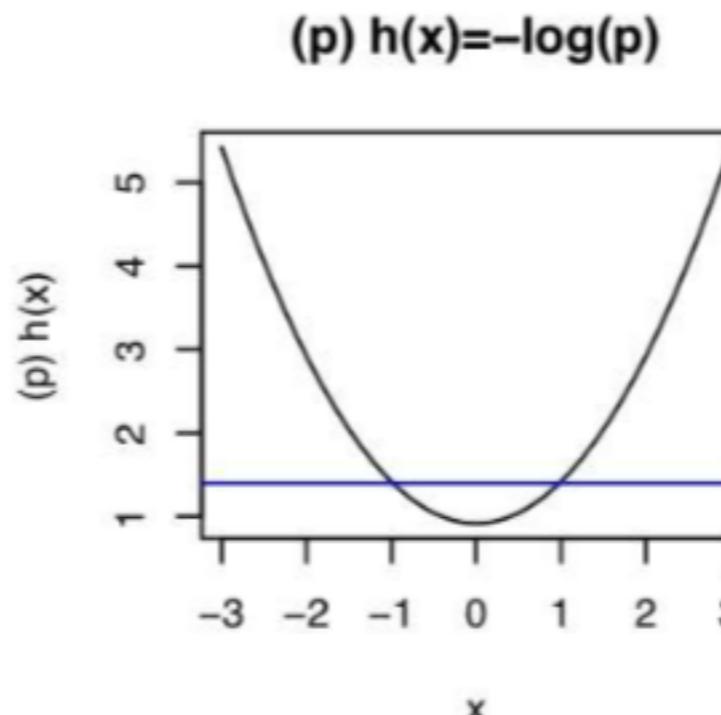
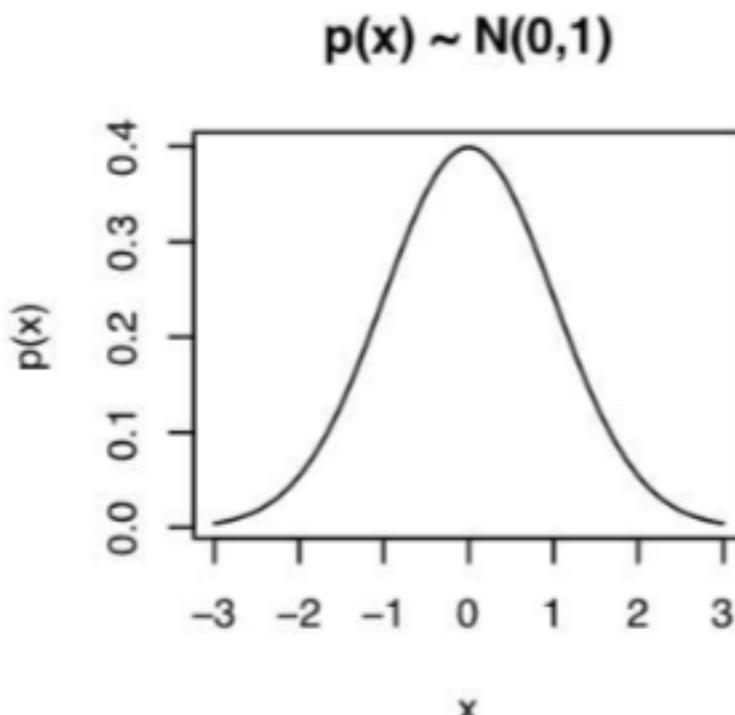
$$h(x) = -\log_2 \left(\frac{1}{6} \right) \cong 2.6$$

이산확률분포에 대한 평균정보량

$$H[x] = - \sum_x p(x) \log_2 p(x)$$

- Entropy란?

- 하나의 계(system)가 가지는 평균 정보량.



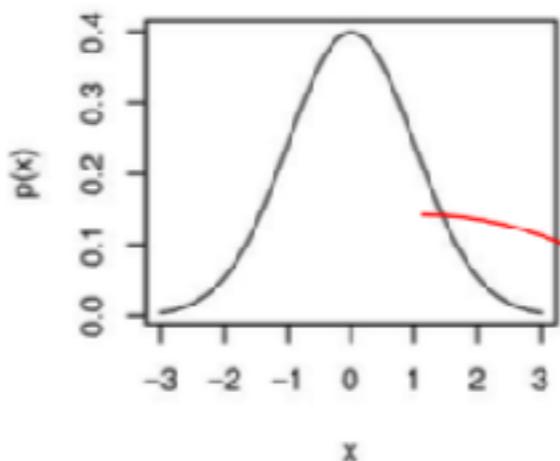
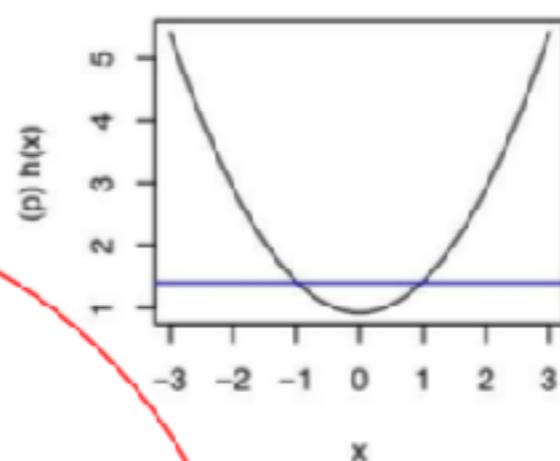
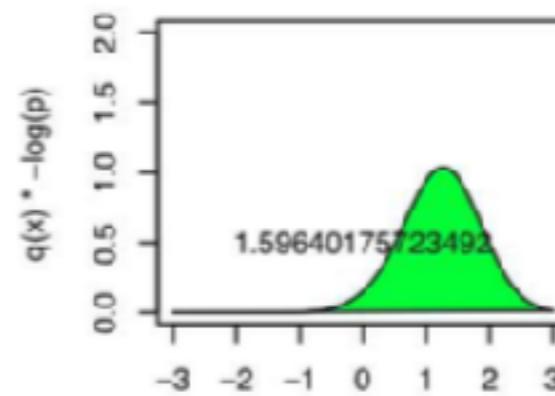
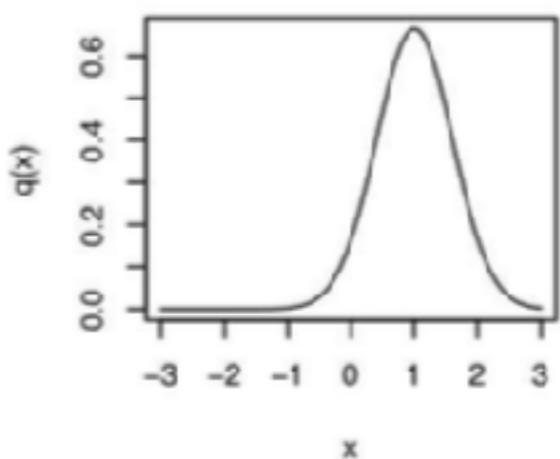
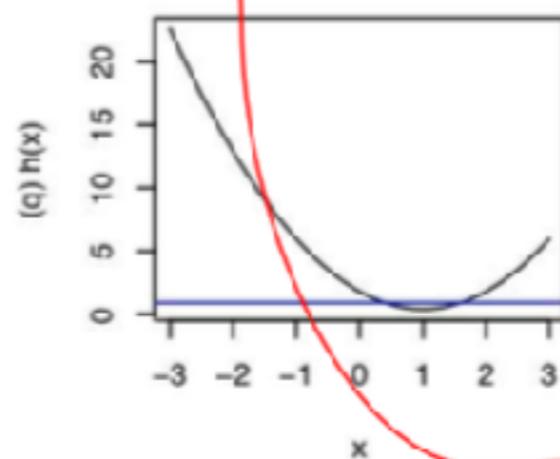
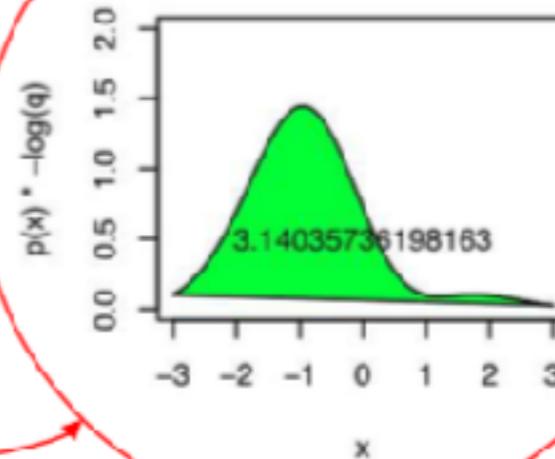
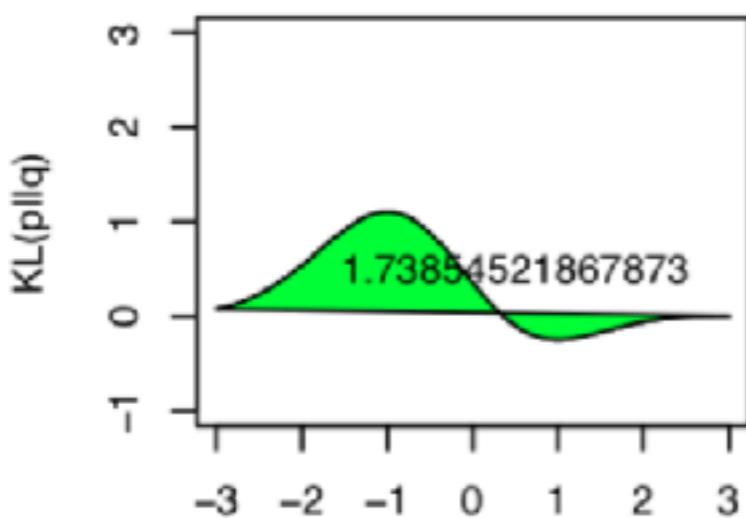
KL-divergence

$$KL(p||q) = - \int p(\mathbf{x}) \ln q(\mathbf{x}) d\mathbf{x} - \left(- \int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x} \right) = - \int p(\mathbf{x}) \ln \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} d\mathbf{x}$$

P, Q의 평균정보량

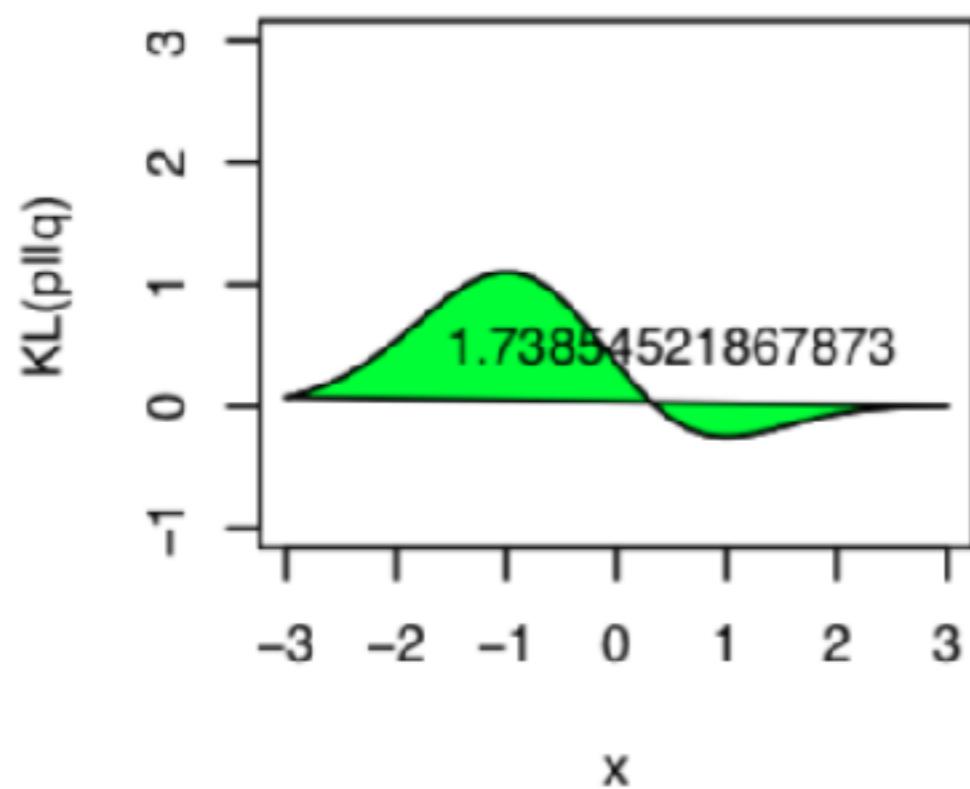
P의 평균정보량

- 정의
 - P라는 확률 분포로 부터 발생한 데이터를,
 - Q라는 확률 분포에서 나왔다고 가정했을 경우
- 이로 인해 발생되는 추가 정보량을 KL-divergence 이라고 한다… 역시 어렵다.

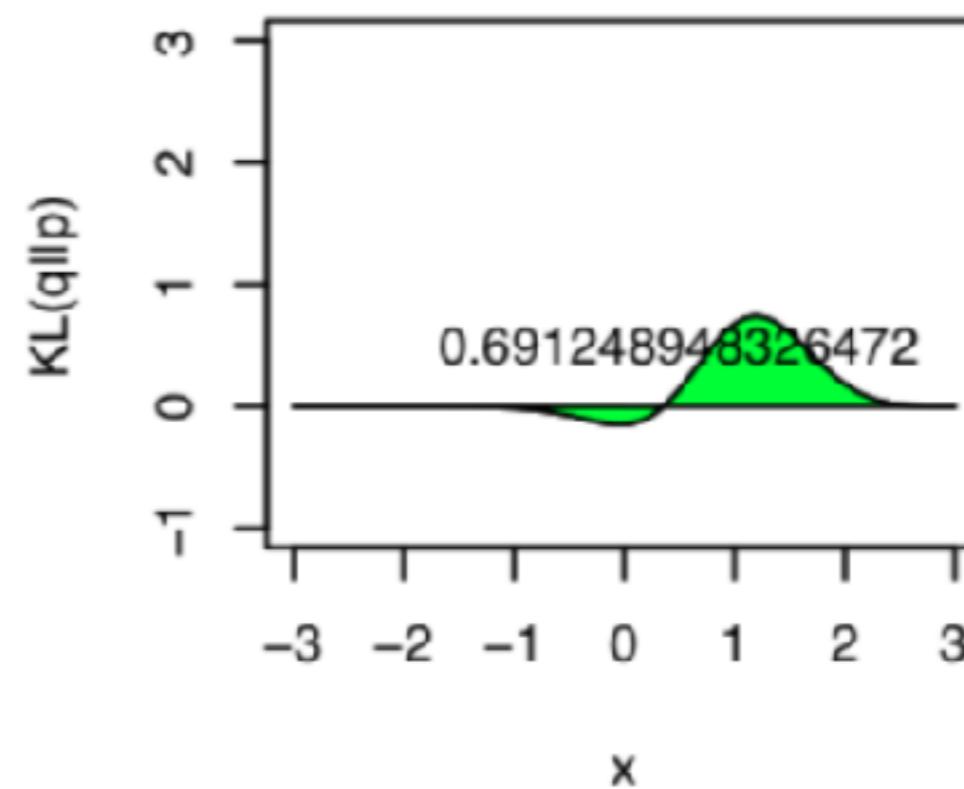
$p(x) \sim N(0, 1)$  $(p) h(x) = -\log(p)$  $R(x) = q(x) * -\log(p)$  $q(x) \sim N(1, 0.6)$  $(q) h(x) = -\log(q)$  $R(x) = p(x) * -\log(q)$  $KL(p||q)$ 

$$KL(p||q) = - \int p(\mathbf{x}) \ln q(\mathbf{x}) d\mathbf{x} - \left(- \int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x} \right)$$

KL(p||q)



KL(q||p)



Asymmetric

하나의 이벤트 집합에서 P, Q 두개의 확률분포에 대한 평균정보량을

크로스 엔트로피로 함.

Cross entropy

From Wikipedia, the free encyclopedia

In information theory, the cross entropy between two probability distributions p and q over the same underlying set of events measures the average number of bits needed to identify an event drawn from the set, if a coding scheme is used that is optimized for an "unnatural" probability distribution q , rather than the "true" distribution p .

The cross entropy for the distributions p and q over a given set is defined as follows:

$$H(p, q) = \mathbf{E}_p[-\log q] = H(p) + D_{\text{KL}}(p\|q),$$

$$KL(p||q) = - \int p(\mathbf{x}) \ln q(\mathbf{x}) d\mathbf{x} - \left(- \int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x} \right)$$

p, q의 크로스 엔트로피 p의 엔트로피

For discrete p and q this means

$$H(p, q) = - \sum_x p(x) \log q(x).$$

- 이걸 어디다 쓸까?
 - 우리가 원래의 분포 P 를 모르는 상태에서 샘플은 P 로부터 얻어진 상황이라면,
 - 어떤 Q 라는 분포 함수를 도입하여 마치 이걸 P 인양 막 쓴다고 하자.
 - 그럼 이런 상황으로 인해 발생되는 오차율을 KL 값을 이용하여 상대 비교가 가능함.
 - 예를 들어 Q_1, Q_2 를 가정하고 각각 KL을 P 에 대해 구해보니 Q_1 이 더 작다.
 - 그러면 Q_1 이 Q_2 보다 P 에 더 가까운 모양이라고 고려할 수 있다.
- 단순한 면적 비교인가?
 - 그건 아니다. 두 함수 사이의 면적차 비율을 최소화하는 방식과는 차이가 있다.
 - 시간이 된다면 몇몇 함수를 도식화해서 확인해보자.
 - P 확률 함수에서 높은 확률을 가지는 지점을 잘 근사해야 KL 값이 작아진다.

이제 우리는 이 모델에서의 inference에 대해 생각해 볼 수 있다. 목표는 관찰된 데이터로부터 숨겨진 변수를 잘 추론(infer)하는 것이다. 곧, posterior $p(z|x)$ 를 계산하는 것과 일치 한다. Bayes에 따르면:

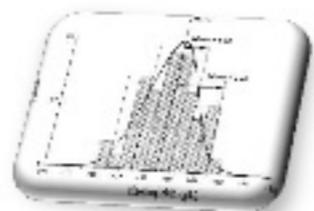
$$p(x, z) = p(x|z)p(z)$$

$$\frac{\text{Posterior}}{\text{Evidence}} = \frac{\text{Likelihood}}{\text{Prior}}$$
$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}.$$

우리의 variational posterior $q(z|x)$ 가
진짜 posterior $p(z|x)$ 를 얼마나 잘 근사하는지 어떻게 알 수 있을까?

근사추론의 방법

$$P(X=0) = q, P(X=1) = p, 0 \leq p \leq 1, q = 1 - p$$



이항분포(Binomial Dist.)

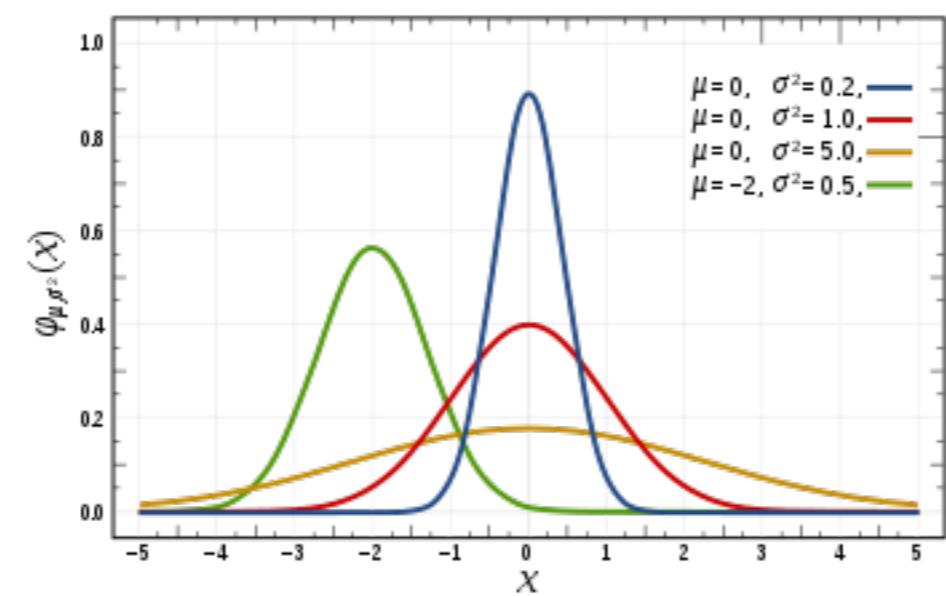
- 베르누이 시행

- 어떤 시행의 결과 성공과 실패로 나타난다.
- 성공의 확률 p ($0 < p < 1$)
- 확률 변수 X 의 실현값은 성공이면 1, 실패면 0
- Ex) 공정한 동전을 던져 앞면이 나오면 성공
 - Bernoulli($p=0.5$) = $p^x(1-p)^{1-x} = 0.5^x(1-0.5)^{1-x}$

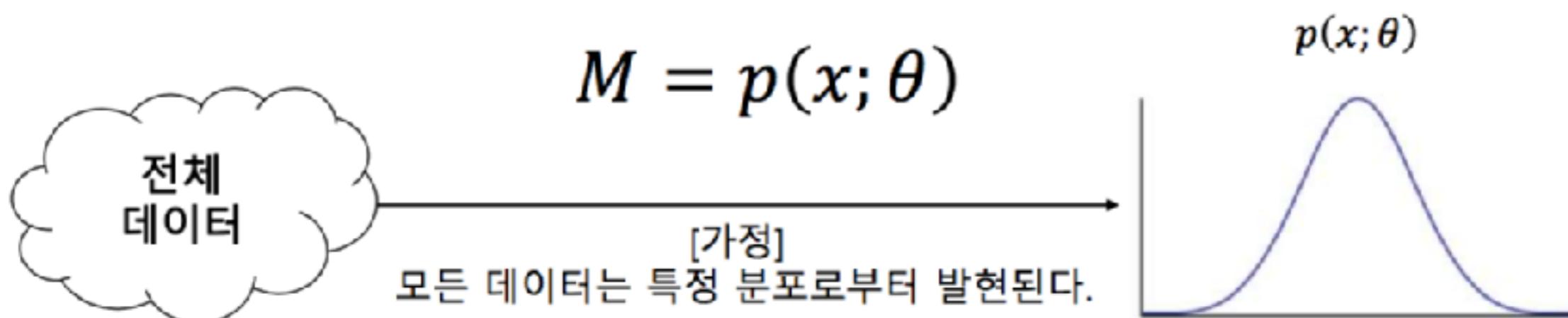
- iid(Independent & Identically)

- 모수(Parameter) : 분포함수의 특징을 결정 짓는 값.
 - 앞선 베르누이 시행에서는 확률값 p
- 동일한 모수를 갖는 확률변수의 실험을 독립적으로 실행하는 것





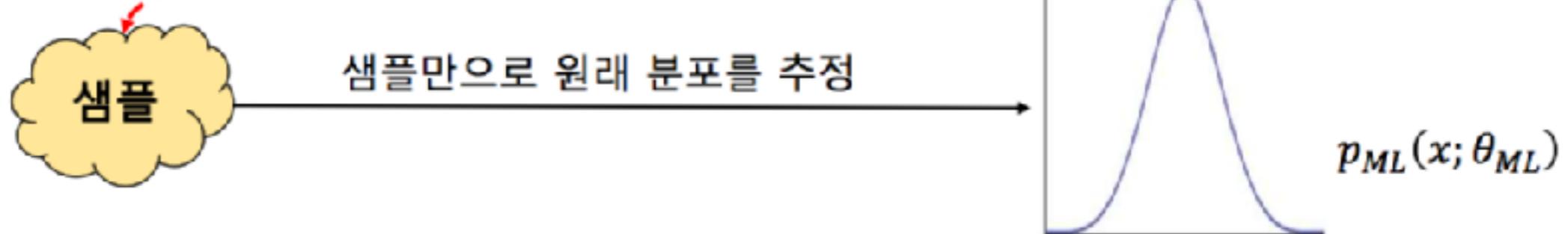
MLE (Maximum Likelihood Estimation)



샘플 또한 $p(x)$ 로부터 발현된 데이터이므로,
이 성질을 이용하여 적절한 L 함수를 설계한다.

$$L(\theta; x) = \operatorname{argmax}_{\theta} p(D; \theta)$$

이런 이유로 likelihood라는 이름이 생겼다.



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

Likelihood **Prior**
Posterior **Evidence**

$$p(z|x) \sim q_{\lambda}(z|x)$$

Variational inference 가 $q_{\lambda}(z|x)$ 계열의 분포를 근사할 수 있다.

Variational parameter λ 는 분포 계열을 구별할 수 있게 한다. 예를 들어, q 가 Gaussian 이라면, 각 datapoint x_i 에 대한 숨겨진 변수의 평균과 분산이 된다. $\lambda_{x_i} = (\mu_{x_i}, \sigma_{x_i}^2)$

우리의 variational posterior $q(z|x)$ 가 진짜 posterior $p(z|x)$ 를 얼마나 잘 근사하는지 어떻게 알 수 있을까? 우리는 q 로 p 를 근사할 때 정보 손실을 측정하는 Kullback-Leibler divergence 를 사용한다.

$$KL(q_\lambda(z|x) || p(z|x)) = \\ \mathbf{E}_q[\log q_\lambda(z|x)] - \mathbf{E}_q[\log p(x, z)] + \log p(x)$$

$$D_{KL}(q(z) || p(z|x)) = \int q(z) \log \frac{q(z)}{p(z|x)} dz \\ = \int q(z) \log \frac{q(z)p(x)}{p(x|z)p(z)} dz \\ = \int q(z) \log \frac{q(z)}{p(z)} dz + \int q(z) \log p(x) dz - \int q(z) \log p(x|z) dz \\ = D_{KL}(q(z) || p(z)) + \log p(x) - E_{z \sim q(z)} [\log p(x|z)]$$

우리의 목표는 divergence 를 최소화하는 λ 를 찾는 것이다. 따라서 최적의 근사 posterior 는 다음과 같다.

$$q_{\lambda}^*(z|x) = \arg \min_{\lambda} KL(q_{\lambda}(z|x) || p(z|x)).$$

$$\mathbf{E}_q[\log q_{\lambda}(z|x)] - \mathbf{E}_q[\log p(x, z)] + \log p(x)$$

왜 이 값을 바로 계산하는게 불가능한가? 귀찮은 evidence $p(x)$ 가 식에 나타나기 때문이다. 위에서 논의했듯이 이 값은 다루기가 매우 힘들다.

To make this notion precise mathematically, we are aiming maximize the probability of each X in the training set under the entire generative process, according to:

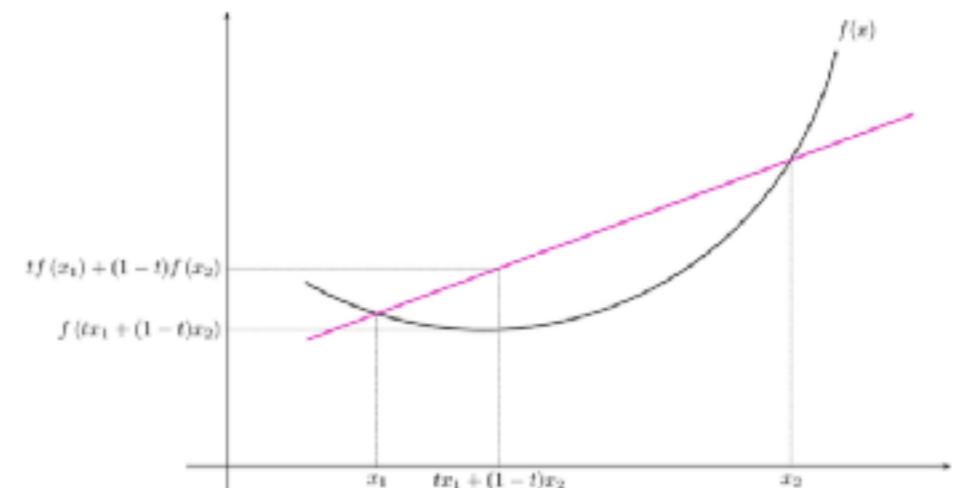
$$P(X) = \int P(X|z; \theta) P(z) dz. \quad (1)$$

The evidence lower bound

- We actually can't minimize the KL divergence exactly, but we can minimize a function that is equal to it up to a constant. This is the **evidence lower bound** (ELBO).
- Recall Jensen's inequality as applied to probability distributions. When f is concave,

$$f(\mathbb{E}[X]) \geq \mathbb{E}[f(X)]. \quad (7)$$

$$\begin{aligned}\log p(x) &= \log \int_z p(x, z) \\ &= \log \int_z p(x, z) \frac{q(z)}{q(z)} \\ &= \log \left(\mathbb{E}_q \left[\frac{p(x, Z)}{q(z)} \right] \right) \\ &\geq \mathbb{E}_q[\log p(x, Z)] - \mathbb{E}_q[\log q(Z)].\end{aligned}$$



다룰 수 있는 variational inference 를 위해선 하나의 조건이 더 필요하다. 아래의 함수를 봄보자:

$$ELBO(\lambda) = \mathbb{E}_q[\log p(x, z)] - \mathbb{E}_q[\log q_\lambda(z|x)].$$

이 식을 Kullback-Leibler divergence 와 합치고 다음과 같이 evidence 를 다시 쓸 수 있음을 확인하자.

$$ELBO(\lambda) = \mathbf{E}_q[\log p(x, z)] - \mathbf{E}_q[\log q_\lambda(z|x)].$$

$$KL(q_\lambda(z|x)||p(z|x)) = \mathbf{E}_q[\log q_\lambda(z|x)] - \mathbf{E}_q[\log p(x, z)] + \log p(x)$$

우리는 q 와 그에 대한 **lambda**를 바꾸고 다루고 있는데,

$$\log p(x) = ELBO(\lambda) + KL(q_\lambda(z|x)||p(z|x))$$

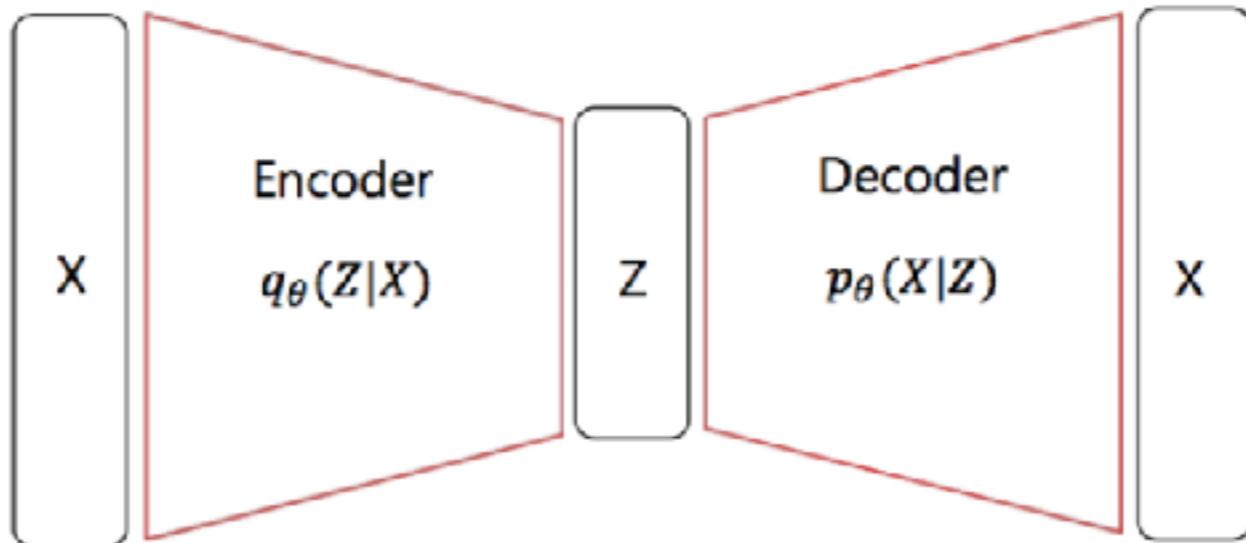
- Notice that $\log p(x)$ does not depend on q . So, as a function of the variational distribution, minimizing the KL divergence is the same as maximizing the ELBO.

ELBO 를 최대화하는 것과 동치임을 말한다. 요약하자면 Evidence Lower BOund (ELBO) 가 posterior inference 근사를 가능하게 한다. 우리는 더 이상 Kullback-Leibler divergence 를 최소화하기 위해 시간을 쓸 필요가 없다. 대신, 우리는 ELBO 를 최대화하므로써 계산 가능한 동치의 일을 수행한다. 좋은 posterior 근사를 얻는 방법으로..

VAE 에선 오직 local latent variable 만이 존재한다 (서로 다른 두 datapoint 가 latent z 를 공유하지 않는다). 따라서 우리는 ELBO 를 단일 datapoint 단위의 합으로 계산한다. 이를 통해 우리는 λ 에 대해 stochastic gradient descent 를 사용할 수 있다. VAE 에서 단일 datapoint에 대한 ELBO 는 다음과 같다. **minibatch..**

$$ELBO_i(\lambda) = E_{q_\lambda(z|x_i)}[\log p(x_i|z)] - KL(q_\lambda(z|x_i)||p(z)).$$

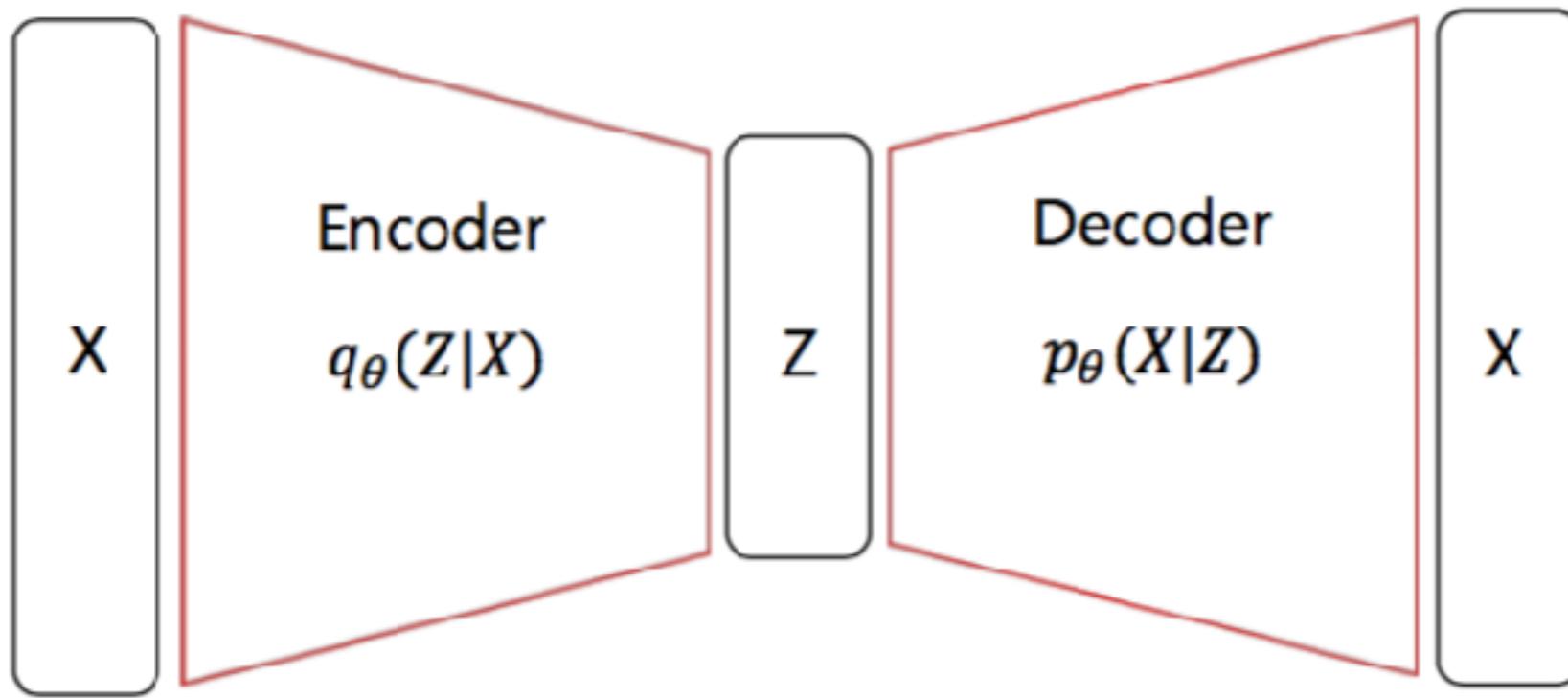
뉴럴넷



이제 뉴럴넷 관점과 연결해보자. 마지막 과정은 근사 posterior $q_\theta(z|x, \lambda)$ 를 데이터 x 를 인풋으로 받고 파라미터 λ 를 아웃풋으로 내보내는 inference network(혹은 encoder)로 파라미터화 하는 것이다. 또한 likelihood $p(x|z)$ 를 latent variables를 인풋으로 받고 데이터 분포 $p_\phi(x|z)$ 의 파라미터를 아웃풋으로 내보내는 generative network(혹은 decoder)로 파라미터화 한다. Inference 와 generative network 는 각각 파라미터 θ 와 ϕ 를 갖는다. 파라미터들은 뉴럴넷 상에서 weight 과 biase 가 된다. 우리는 stochastic gradient descent 를 통해 이를 최적화하여 ELBO 를 최대화 한다. (global latent variable 이 없기 때문에 데이터를 minibatch 로 다루는 것이 더 정결하다) ELBO 에 inference 와 generative network 의 파라미터를 더해서 쓰면 다음과 같다.

$$ELBO_i(\theta, \phi) = E_{q_\theta(z|x_i)}[\log p_\phi(x_i|z)] - KL(q_\theta(z|x_i)||p(z)).$$

이 evidence lower bound 는 VAE 의 loss function 의 역으로써, 우리가
뉴럴넷 관점에서 논의한 내용이다; $ELBO_i(\theta, \phi) = -l_i(\theta, \phi)$. 우리는
원리를 통해 이 결론에 도달하기 위해 확률 모델과 approximate
posterior inference 를 사용하였다. 우리는 여전히 Kullback-Leibler
divergence 를 regularizer 관점, 또는 재생성 손실 관점에서 likelihood 기
대값으로 생각할 수 있다. 하지만 확률 모델을 통한 설명이 왜 이 부분이
존재하는지를 분명하게 보여준다: approximate posterior $q_\lambda(z|x)$ 와
posterior $p(z|x)$ 사이의 Kullback-Leibler divergence 를 최소화하는 일.



문제점

① $q_\theta(Z|X)$ 를 계산 할 수 없음. Intractable함.

→ $q_\theta(Z|X)$ 라는 확률분포를 정의해주고 KL-Divergence를 이용해 근사 시켜준다(Variational Inference).

② Backpropagation을 하려면 모델이 미분 가능 해야 하는데 $q_\theta(Z|X)$ 는 미분 불가능하다.

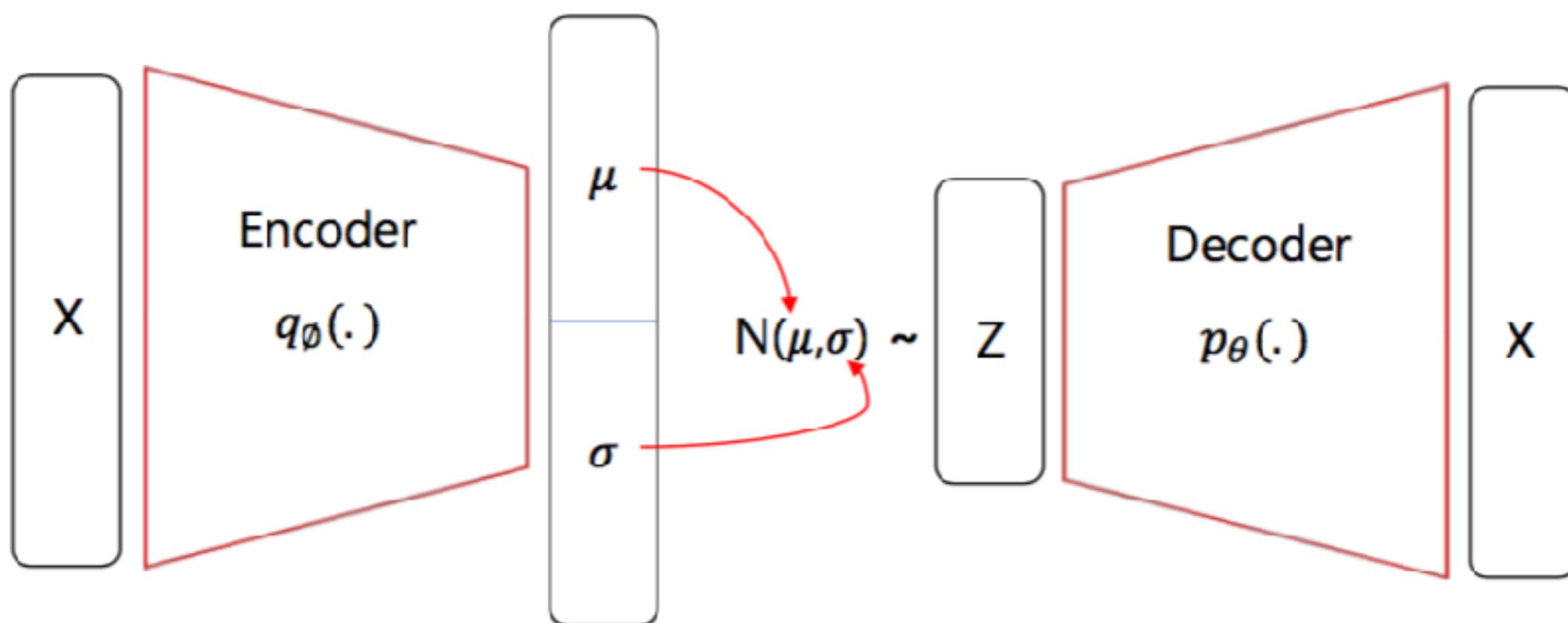
→ Reparameterization Trick을 이용해 해결, 논문의 first contribution.

문제점

① $q_\theta(Z|X)$ 를 계산 할 수 없음. Intractable함. 아래의 cs231n slide 참조.

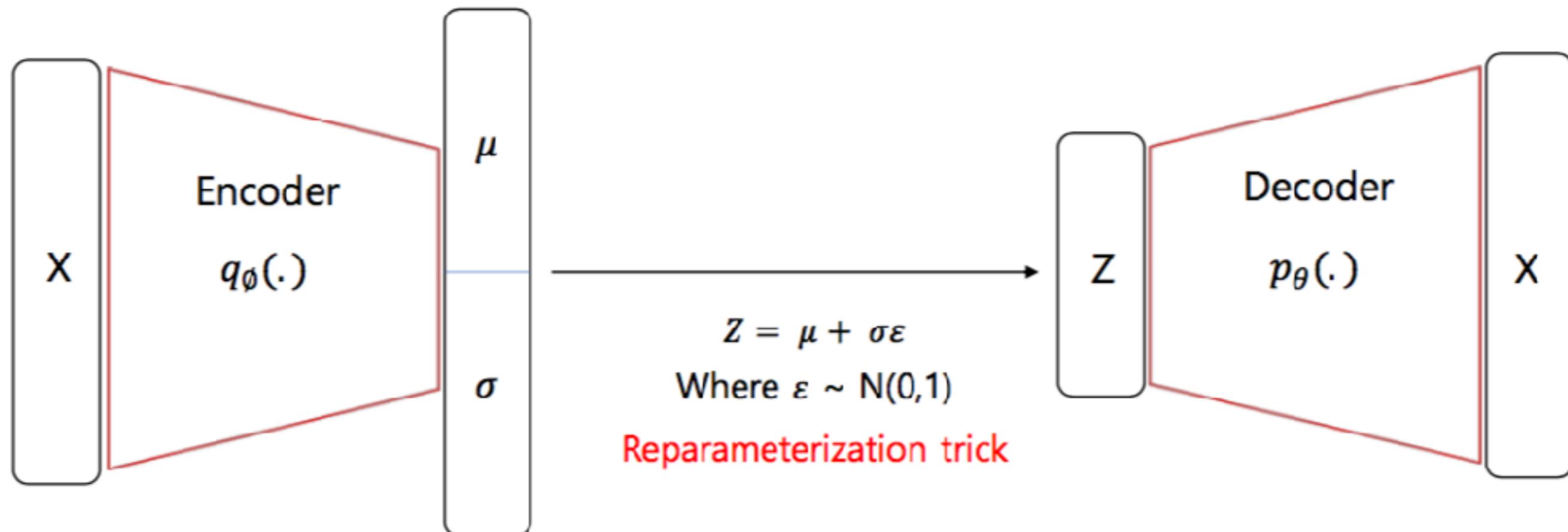
→ $q_\theta(X|Z)$ 라는 확률분포를 정의해주고 KL-Divergence를 이용해 근사 시켜준다(Variational Inference).

$q_\theta(Z|X)$ 를 Gaussian distribution으로 가정하면, 우리의 네트워크는 아래와 같이 된다.



문제점

- ② Backpropagation을 하려면 모델이 미분가능 해야 하는데 $q_\theta(X|Z)$ 는 미분 불가능하다.
→ Reparameterization Trick을 이용해 해결, 논문의 first contribution.



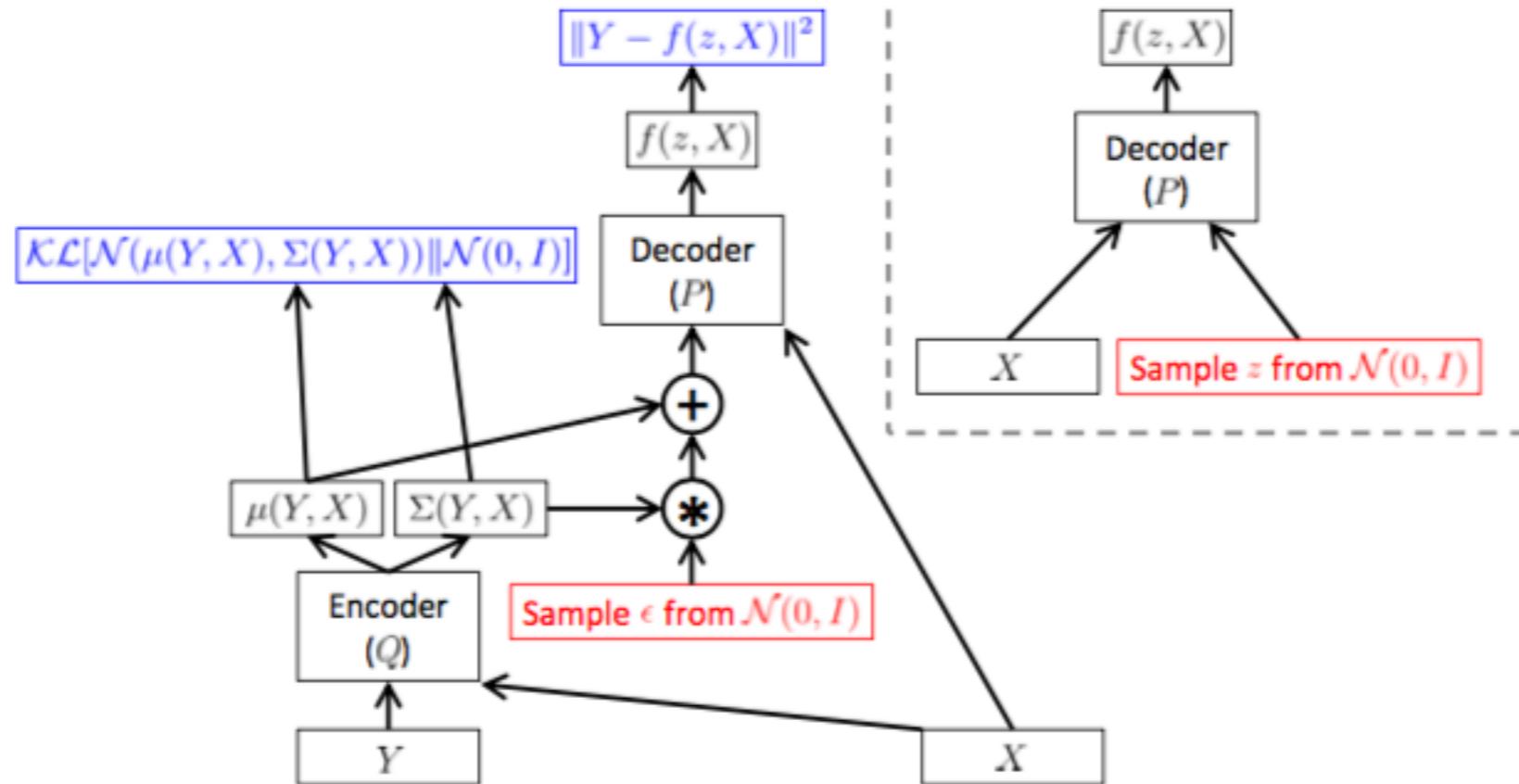


Figure 6: Left: a training-time conditional variational autoencoder implemented as a feedforward neural network, following the same notation as Figure 4. Right: the same model at test time, when we want to sample from $P(Y|X)$.

MLE로 parameter를 추정하는데,

$\log(P_\theta(X)) = L^{(a)}(\theta, \emptyset; X) + D_{KL}(q_\emptyset(Z|X) || P_\theta(Z|X)) \geq L^{(a)}(\theta, \emptyset; X)$ 이므로 Lower bound인 $L^{(a)}(\theta, \emptyset; X)$ 을 최대화 하면 $\log(P_\theta(X))$ 값도 최대가 된다고 생각할 수 있다. 따라서 우리의 목표는 다음과 같이 다시 작성될 수 있다.

$$\text{Argmax}_\theta(\log(P_\theta(X))) \leftrightarrow \text{Argmax}_{\theta, \emptyset}(L^{(a)}(\theta, \emptyset; X))$$

앞서본 것처럼 ELBO 최대화해서.. 추정

$$\text{Argmax}_{\theta, \emptyset}(L^{(a)}(\theta, \emptyset; X)) = \text{Argmax}_{\theta, \emptyset}\left(\underbrace{\int_Z q_\emptyset(Z|X) \times \log(P_\theta(X|Z))}_{(a)} - \underbrace{D_{KL}(q_\emptyset(Z|X) || P_\theta(Z))}_{(b)}\right)$$

위의 식에서 (b)는 $q_\emptyset(Z|X)$ 가 Gaussian distribution, $P_\theta(Z)$ 가 Normal distribution이라고 할 때

다음과 같이 계산 가능하다. $D_{KL}(q_\emptyset(Z|X) || P_\theta(Z)) = \frac{1}{M} \sum_{i=1}^M \left[\frac{1}{2} \sum_{j=1}^D \{1 - \log(\sigma_{i,j}^2) + \mu_{i,j}^2 + \sigma_{i,j}^2\} \right]$

Where D는 Data의 Dimension(MNIST의 경우 784), M은 데이터의 갯수

(b)를 계산할 때,
실제 코드에 사용하는 식

$$Argmax_{\theta, \emptyset} \left(L^{(a)}(\theta, \emptyset; X) \right) = Argmax_{\theta, \emptyset} \left(\underbrace{\int_Z q_{\emptyset}(Z|X) \times \log(P_{\theta}(X|Z))}_{(a)} - \underbrace{D_{KL}(q_{\emptyset}(Z|X) || P_{\theta}(Z))}_{(b)} \right)$$

(a)의 경우, 아래와 같은 방법으로 계산하면 결론적으로 Cross_entropy와 같은 결과를 얻게 된다.

$$\int_Z q_{\emptyset}(Z|X) \times \log(P_{\theta}(X|Z)) \approx \frac{1}{L} \sum_{Z_l=Z_1}^{Z_L} \log(P_{\theta}(X|Z_l)) \dots \text{by Monte carlo technique}$$

편의를 위해 $L = 10$ 이라고 가정하면

준식 = $\log(P_{\theta}(X|Z_1))$ 가 된다. X 의 Dimension을 D 라고 가정하고, data의 수를 M 이라 가정했을 시 아래와 같이 전개할 수 있다.

$$\log(P_{\theta}(X|Z_1)) = \frac{1}{M} \sum_{i=1}^M \log \left\{ \prod_{j=1}^D P_{\theta}(X_{i,j}|Z_1) \right\} = \frac{1}{M} \sum_{i=1}^M \sum_{j=1}^D \log\{P_{\theta}(X_{i,j}|Z_1)\}$$

$P_{\theta}(X_{i,j}|Z_1)$ 을 Bernoulli distribution이라 가정했을 시 위 식은 아래와 같이 전개 가능하다.

$$\begin{aligned} &= \frac{1}{M} \sum_{i=1}^M \sum_{j=1}^D \log P_{i,j}^{X_{i,j}} (1 - P_{i,j})^{(1-X_{i,j})} \\ &= \frac{1}{M} \sum_{i=1}^M \sum_{j=1}^D \{X_{i,j} \log P_{i,j} + (1 - X_{i,j}) \log(1 - P_{i,j})\} \dots \text{Cross Entropy Equation.} \end{aligned}$$

따라서 앞의 두 슬라이드에서 계산했듯이

$$\text{Argmax}_{\theta, \emptyset} \left(L^{(a)}(\theta, \emptyset; X) \right) = \text{Argmax}_{\theta, \emptyset} \left(\int_Z q_{\emptyset}(Z|X) \times \log(P_{\theta}(X|Z)) - D_{KL}(q_{\emptyset}(Z|X) || P_{\theta}(Z)) \right)$$

이 문제는 아래의 값의 최대 값을 구하는 문제와 동치이게 된다.

$$L^{(a)}(\theta, \emptyset; X) = \frac{1}{M} \sum_{i=1}^M \sum_{j=1}^D \{ X_{i,j} \log P_{i,j} + (1 - X_{i,j}) \log(1 - P_{i,j}) \} - \frac{1}{L} \sum_{i=1}^L \left[\frac{1}{2} \sum_{j=1}^J \{ 1 - \log(\sigma_{i,j}^2) + \mu_{i,j}^2 + \sigma_{i,j}^2 \} \right]$$

Fully convolutional network에서는 Gradient Descent Algorism을 사용하므로

$$-L^{(a)}(\theta, \emptyset; X) = -\frac{1}{M} \sum_{i=1}^M \sum_{j=1}^D \{ X_{i,j} \log P_{i,j} + (1 - X_{i,j}) \log(1 - P_{i,j}) \} + \frac{1}{L} \sum_{i=1}^L \left[\frac{1}{2} \sum_{j=1}^J \{ 1 - \log(\sigma_{i,j}^2) + \mu_{i,j}^2 + \sigma_{i,j}^2 \} \right]$$

값을 최소화 시키는 파라미터들을 학습 시켜주면 된다.

```
def gaussian_encoder(X, n_hidden, n_z, keep_prob):
    w_init = tf.contrib.layers.xavier_initializer()
    input_shape = X.get_shape()

    with tf.variable_scope("encoder_hidden_1", reuse = tf.AUTO_REUSE):
        w1 = tf.get_variable("w1", shape = [input_shape[1], n_hidden], initializer = w_init)
        b1 = tf.get_variable("b1", shape = [n_hidden], initializer = tf.constant_initializer(0.))
        h1 = tf.matmul(X,w1) + b1
        h1 = tf.nn.elu(h1)
        h1 = tf.nn.dropout(h1, keep_prob)

    with tf.variable_scope("encoder_hidden_2", reuse = tf.AUTO_REUSE):
        w2 = tf.get_variable("w2", shape = [n_hidden,n_hidden], initializer = w_init)
        b2 = tf.get_variable("b2", shape = [n_hidden], initializer = tf.constant_initializer(0.))
        h2 = tf.matmul(h1,w2) + b2
        h2 = tf.nn.elu(h2)
        h2 = tf.nn.dropout(h2,keep_prob)

    with tf.variable_scope("encoder_z", reuse = tf.AUTO_REUSE):
        w3 = tf.get_variable("w3", shape = [n_hidden, n_z*2], initializer = w_init)
        b3 = tf.get_variable("b3", shape = [n_z*2], initializer = tf.constant_initializer(0.))
        h3 = tf.matmul(h2,w3) + b3
        mean = h3[:, : n_z]
        std = tf.nn.softplus(h3[:, n_z :]) + 1e-6

    return mean, std
```

```
def Bernoulli_decoder(z, n_hidden, n_out, keep_prob):
    w_init = tf.contrib.layers.xavier_initializer()
    z_shape = z.get_shape()

    with tf.variable_scope("decoder_hidden_1", reuse = tf.AUTO_REUSE):
        w4 = tf.get_variable("w4", shape = [z_shape[1],n_hidden], initializer = w_init)
        b4 = tf.get_variable("b4", shape = [n_hidden], initializer = tf.constant_initializer(0.))
        h4 = tf.matmul(z,w4) + b4
        h4 = tf.nn.elu(h4)
        h4 = tf.nn.dropout(h4,keep_prob)

    with tf.variable_scope("decoder_hidden_2", reuse = tf.AUTO_REUSE):
        w5 = tf.get_variable("w5", shape = [n_hidden,n_hidden], initializer = w_init)
        b5 = tf.get_variable("b5", shape = [n_hidden], initializer = tf.constant_initializer(0.))
        h5 = tf.matmul(h4,w5) + b5
        h5 = tf.nn.elu(h5)
        h5 = tf.nn.dropout(h5, keep_prob)

    with tf.variable_scope("decoder_output", reuse = tf.AUTO_REUSE):
        w6 = tf.get_variable("w6", shape = [n_hidden,n_out], initializer = w_init)
        b6 = tf.get_variable("b6", shape = [n_out], initializer = tf.constant_initializer(0.))
        h6 = tf.matmul(h5,w6) + b6
        h6 = tf.nn.sigmoid(h6)

    return h6
```

```
def Variational_autoencoder(X,n_hidden_encoder,n_z, n_hidden_decoder, keep_prob ):  
    X_shape = X.get_shape()  
    n_output = X_shape[1]  
  
    mean, std = gaussian_encoder(X,n_hidden_encoder, n_z,keep_prob)  
  
    z = mean + std*tf.random_normal(tf.shape(mean,out_type = tf.int32), 0, 1, dtype = tf.float32)  
  
    X_out = Bernoulli_decoder(z,n_hidden_decoder,n_output,keep_prob)  
    X_out = tf.clip_by_value(X_out,1e-8, 1 - 1e-8)  
  
    likelihood = tf.reduce_mean(tf.reduce_sum(X*tf.log(X_out) + (1-X)*tf.log(1- X_out),1))  
    KL_Divergence = tf.reduce_mean(0.5*tf.reduce_sum(1 - tf.log(tf.square(std) + 1e-8) + tf.square(mean) + tf.square(std), 1))  
    Recon_error = -1*likelihood  
    Regularization_error = KL_Divergence  
    ELBO = Recon_error + Regularization_error  
  
    return z ,X_out, Recon_error, Regularization_error, ELBO
```

posterior inference 를 사용하였다. 우리는 여전히 Kullback-Leibler divergence 를 regularizer 관점, 또는 재생성 손실 관점에서 likelihood 기대값으로 생각할 수 있다. 하지만 확률 모델을 통한 설명이 왜 이 부분이

$$-L^{(a)}(\theta, \phi; X) = \underbrace{\int_Z q_\phi(Z|X) \times \log(P_\theta(X|Z))}_{\text{Reconstruction Error}} - \underbrace{D_{KL}(q_\phi(Z|X) || P_\theta(Z))}_{\text{Regularization}}$$

Regularization

주어진 데이터로 부터 Z를 잘 샘플링 했는지를 나타내는 척도

Reconstruction Error

AutoEncoder 관점에서 본 X에 대한 복원오차

Notes on Variational Autoencoders

<http://pyro.ai/examples/vae.html>

Modern applications and data strongly favour **probabilistic modelling**:

- *Noise in the data* and account for our lack of knowledge
- *Non-iid, non-stationary* data.
(independent and identical distributed) data
Non-independent : Markovian
 - Explore and extract the *underlying structure* in the data
 - *Consistency in our beliefs* about the data and systems we study.

$$\text{Posterior } p(z|y) = \frac{\text{Likelihood } p(y|z) \text{ Prior } p(z)}{\int p(y, z) dz}$$

**Marginal likelihood/
Model evidence**

$$E[f] = \text{Sigma}(p(x)f(x))$$

$$E[f] = \text{Integral}(p(x)f(x)dx)$$