Variational Auto Encoder

Variational Auto Encoder,

Auto-Encoding Variational Bayes, 2014

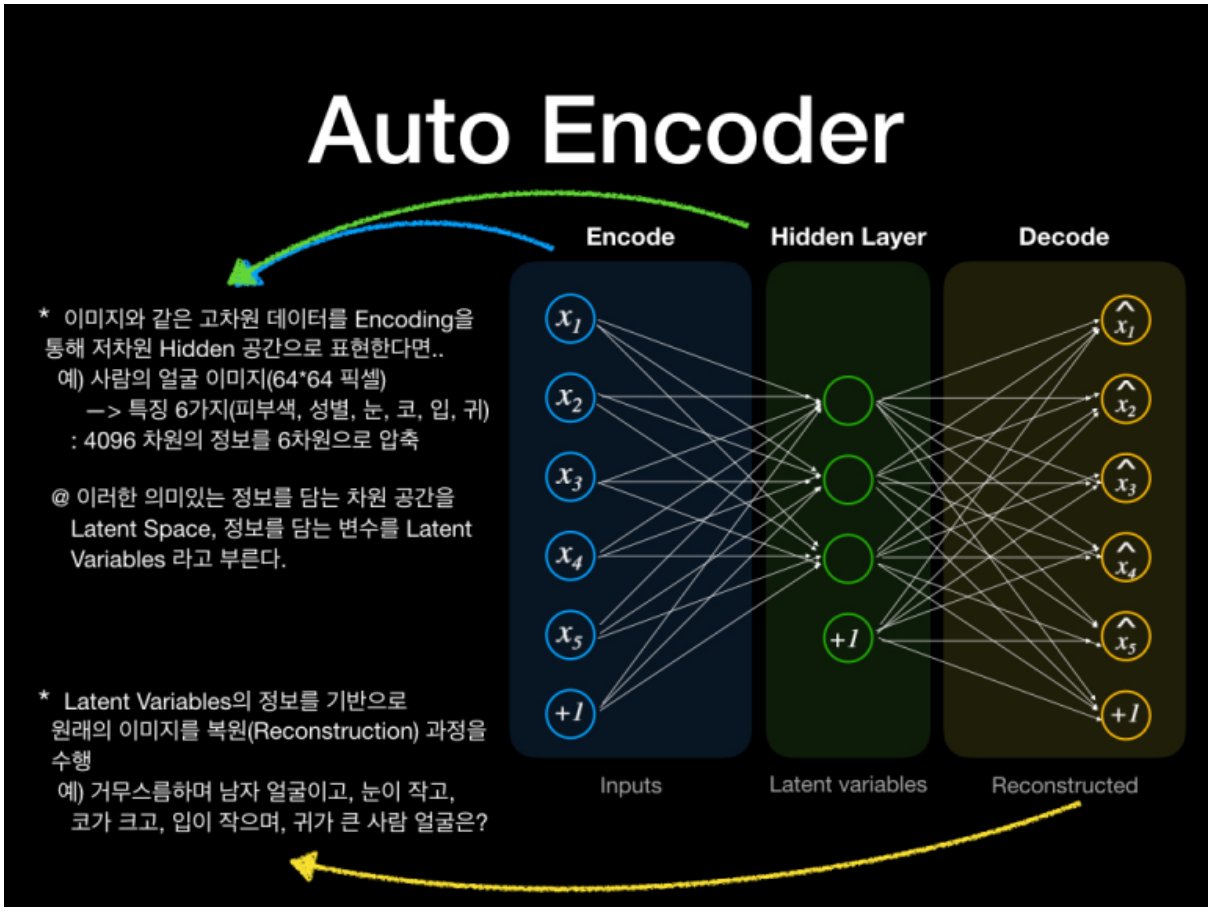
Auto Encoder의 구조는 크게 Encoder와 Decoder 두 파트로 구성된다.

어떤 입력에 대하여 → ‘특징을 추출하여’ → 입력의 압축된 정보를 latent space에 담고, 이 latent space로부터 다시 자기 자신을 추출하는 알고리즘이다.

주로 사용하는 분야로는, Encoder를 통해 ‘정보를 compress’할수도 있고, Decoder을 통해 다시 자기 자신의 정보를 복원하는 등의 역할을 수행할 수 있다.

특징 추출이나, information-compressive한 역할을 수행하는 pca/LDA와도 결은 어느 정도 비슷하다고 할 수 있을 것이다.

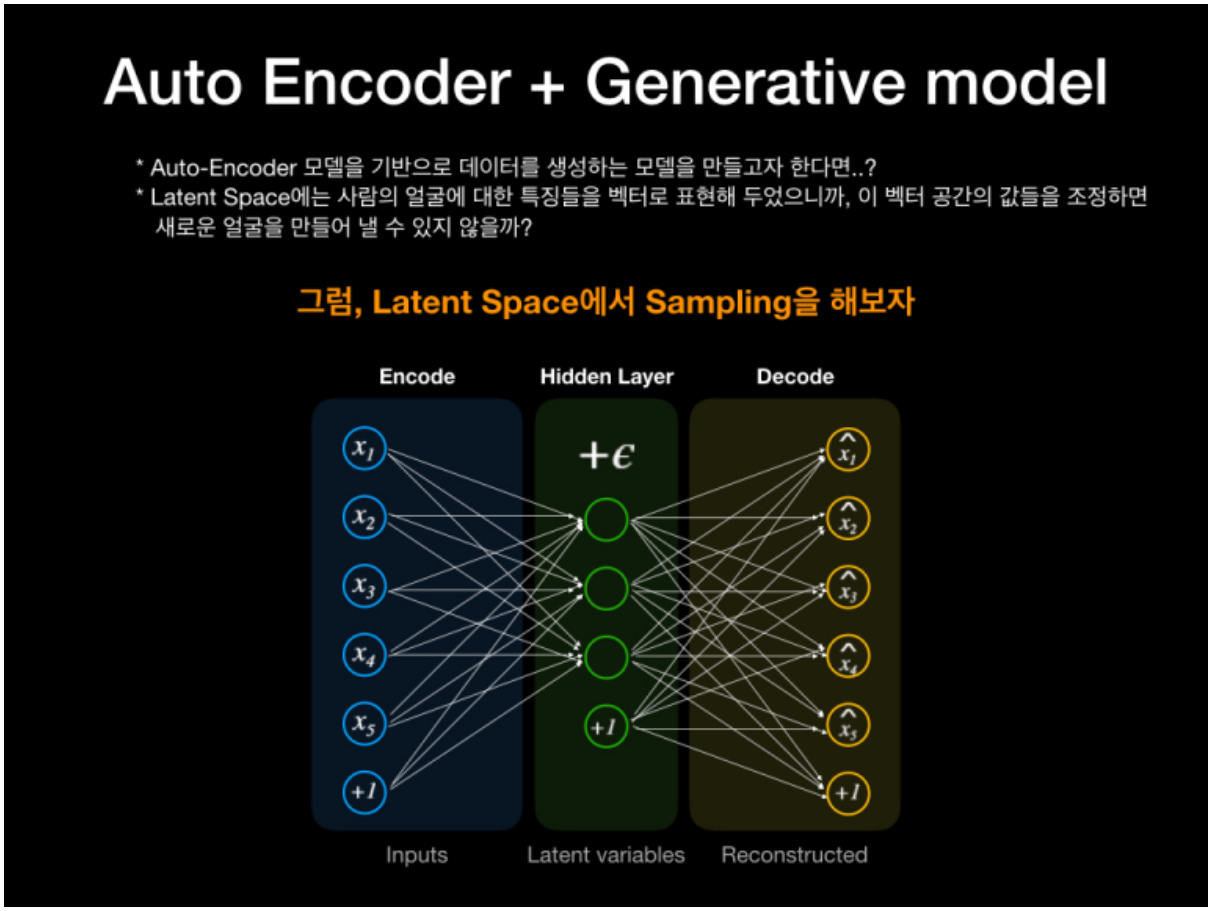
또한, 노이즈가 포함된 입력에 대해, de-noising하는 문제에서 효과적으로 쓰일 수 있다.



Image가 input되었다고 했을 때, 이 때 이미지는 Encoder를 통해 저차원의 Hidden 공간(latent한 공간)으로 압축된다.

예를 들어 피부색, 성별, 눈, 코, 입, 귀라는 6가지 특징으로 압축되었다고 가정한다. 이렇게 의미 있는 정보가 담긴 압축된 공간을 Latent Space, 이 변수들을 Latent Variables이라고 칭한다.

Decoder의 과정은 다시 피부색, 성별, 눈, 코, 입, 귀 의 특징값에 해당하는 새로운 얼굴 이미지를 생성하게 된다.



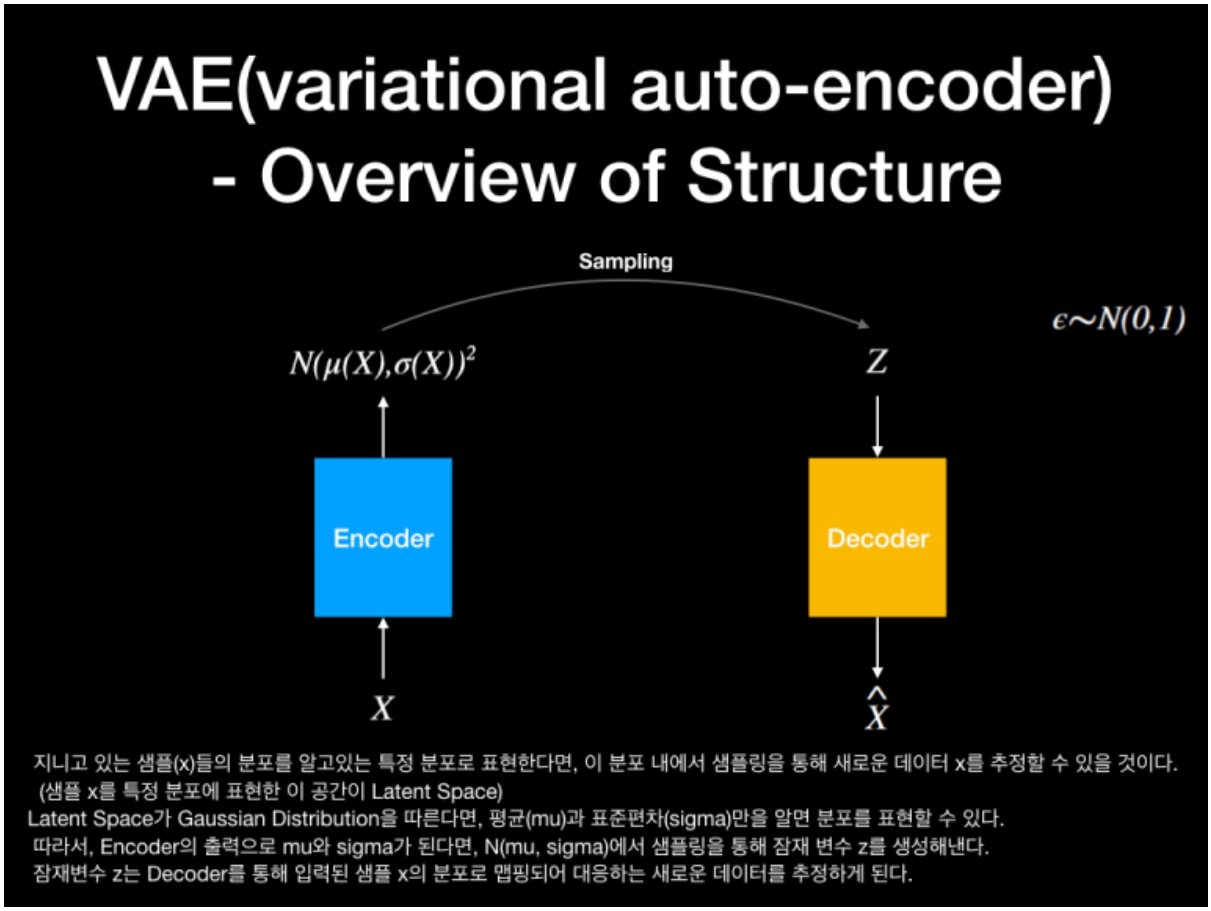
Latent Space에는 입력 데이터들에 대해서 어떤 특징들로써 정보를 함축하고 있다.

그렇다면 **“Latent Space로부터 데이터를 생성해 낼 수는 없을까?”**라는 점에서 VAE의 아이디어가 출발한다.

AE와 VAE의 차이점은 AE는 잠재공간 z에 값을 저장하고 VAE는 확률 분포를 저장하여 (평균, 분산) 파라미터를 생성한다는 점이다.

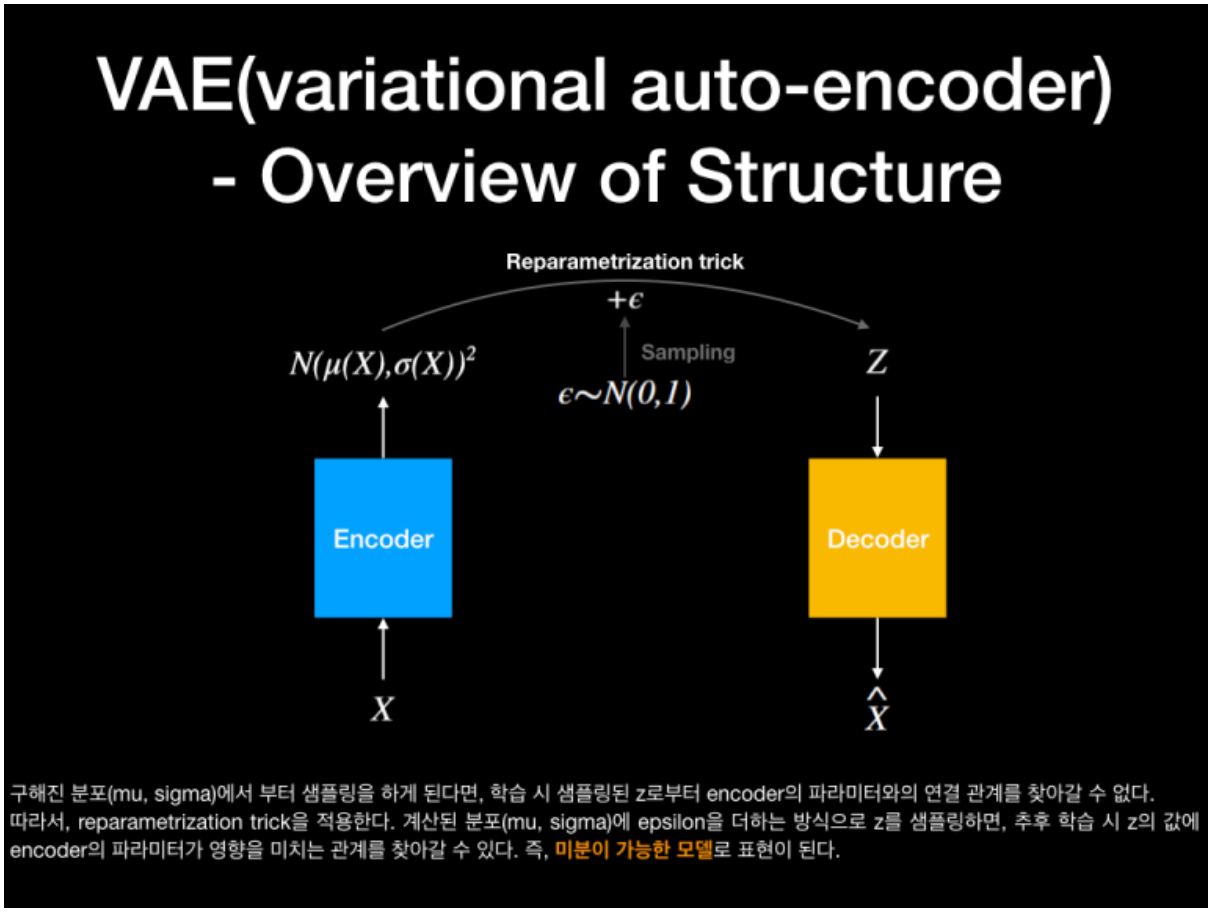
즉, AE는 information Compression에 focus 되어 있다면,

VAE는 Latent space로부터 output generation에 더 집중되어있다.

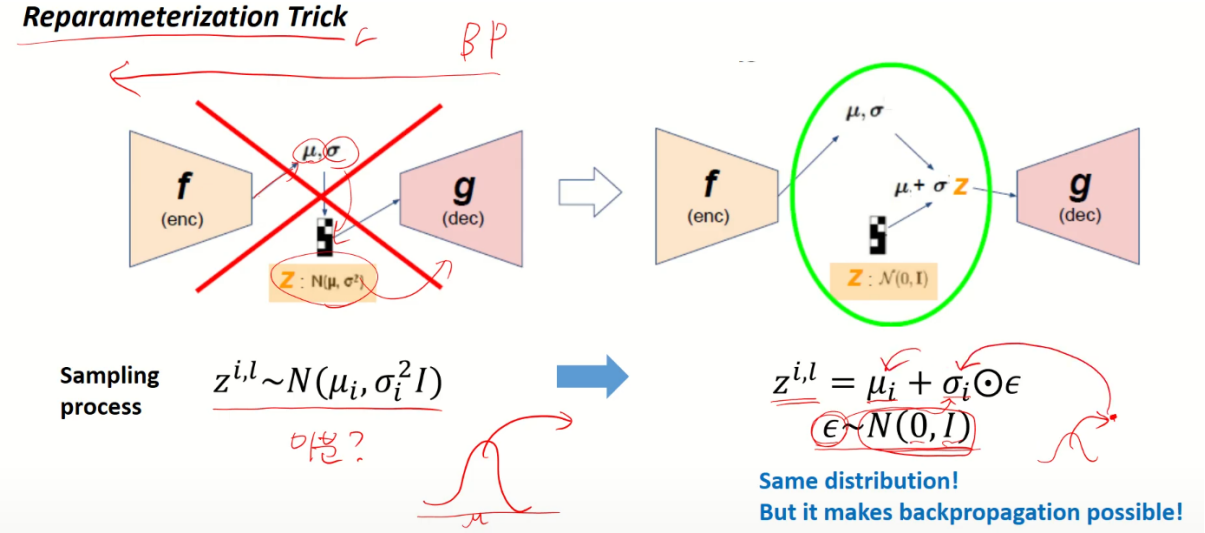


- 이미지 데이터와 같은 경우에는 입력 데이터의 차원이 높고 데이터의 양 또한 많다(담고 있는 정보가 많다)
- 이 때, 데이터의 분포를 사전에 알고 있다면, 샘플링을 통해 새로운 데이터를 추정할 수 있다.
- 이러한 방식으로, 새로운 데이터를 생성함으로써 데이터의 양을 늘리고 다양성을 확보할 수 있을 것이다.
- 이 과정에서, 실제 데이터와 생성된 데이터의 분포가 유사해야 하며 이를 위해, 실제 데이터의 분포와 알고 있는 분포를 매핑하여 생성된 데이터가 실제 데이터와 유사하도록 한다.
-
- AutoEncoder의 Latent Space가 우리가 잘 아는 정규분포를 따른다면, 이 분포는 평균(mu)과 표준편차(sigma)만 구해낸다면 분포를 표현할 수 있다.
- 그러므로 Encoder의 출력이 mu와 sigma가 된다면, 정규분포 N(mu, sigma)에서 샘플링을 통해 잠재 변수 z를 생성할 수 있고,

- ⇒ 생성된 잠재변수 z는 다시 Decoder를 통해 입력된 샘플 x의 분포로 매핑되어 대응하는 새로운 데이터를 추정할 수 있다.
- 즉 input 과정에서 encoder에 input 되었을 경우에 mu와 sigma가 계산되어 정규분포로서 분포가 만들어지고, 이 때 하나의 샘플링이 진행된다.

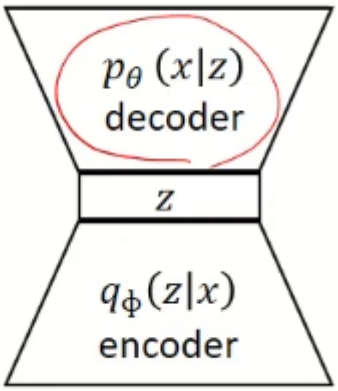


Reparametrization Trick



그러나 샘플링을 하게 되면 무작위성을 띄기 때문에 학습을 할 수 없게 된다.

즉, 이러한 sampling 연산을 미분할 수 없기 때문에 back-propagation을 통한 parameter optimization이 불가능해진다는 문제점을 가진다.



샘플링 연산을

$$p_\theta(x) = z$$

라 하면, 모델 자체는

$$f_\theta(z).$$

로 나타낼 수 있다.

$$p_\theta(x)$$

를 따라서 바로 미분할 수 없기 때문에,

$$q_\phi(z) = x$$

라는 함수를 근사하고, 이를 이용하여 샘플링 연산과 동일한 결과를 얻는 함수를 찾는 것이다.

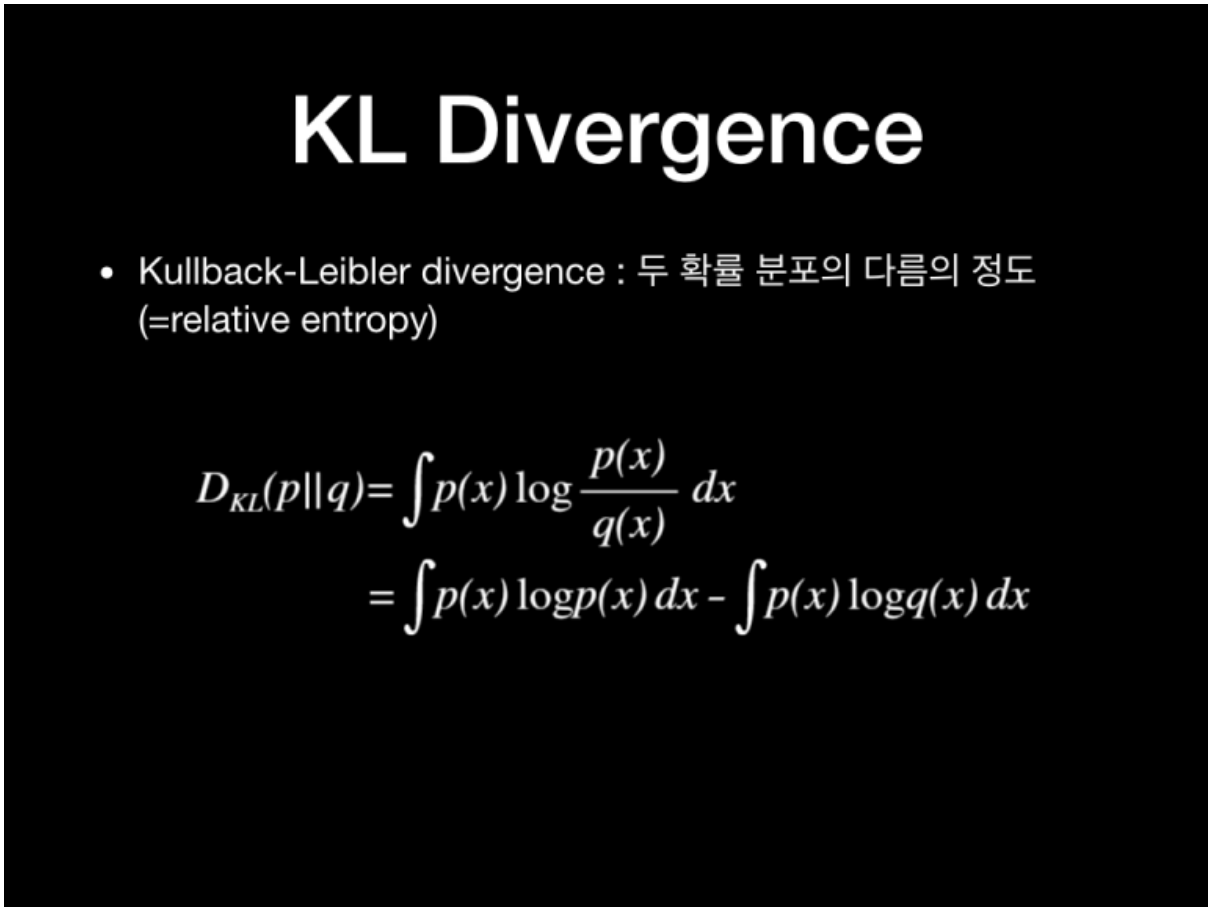
$$q_\phi(z) \simeq f_\theta(z)$$

후자는 랜덤성을 띄기 때문에, 미분이 불가하지만 전자는 미분이 가능하며, 따라서 둘은 거의 동일한 결과를 얻기 때문에 바꿔 쓸 수 있다는 논리인 것이다.

이러한 방법을 reparametrization trick이라 하며, 논문에서는 이와 같은 트릭을 사용한다.

⇒ 이 트릭을 통해 N(mu, sigma) + e, e~N(0,1)와 같이 e를 출력에 더해줌으로써 z 값에 encoder가 미치는 영향 관계를 찾을 수 있게 된다. 이렇게 함으로써, 미분 가능한 모델로 표현이 가능해진다.

KL Divergence



쉽게 말해서, 실제 분포와 내가 예측한 분포와의 차이를 의미한다. 쉽게 말해서, 내가 예측한 분포가 얼마나 '바보같은가'를 의미한다.

KL Divergence

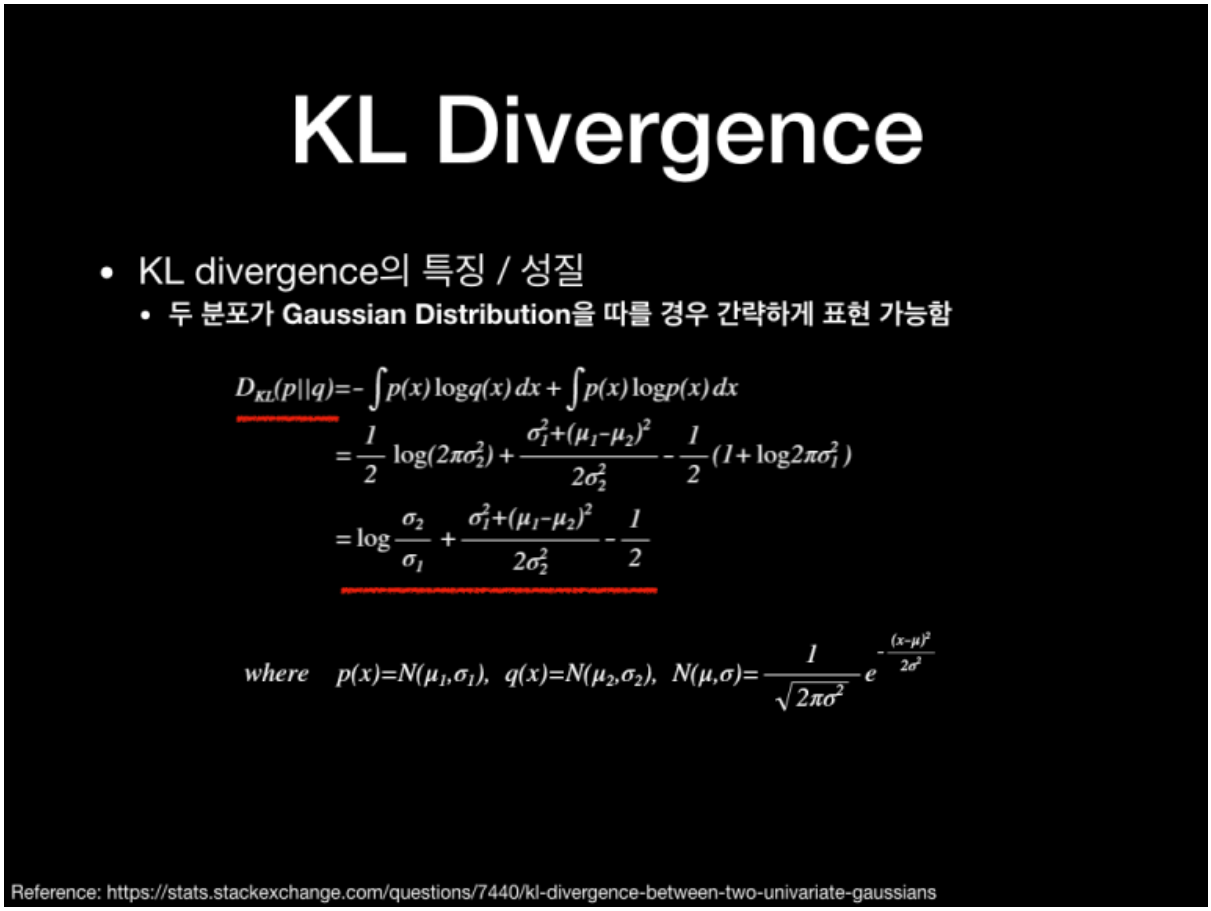
- KL divergence의 특징 / 성질
  - 항상 0 이상의 값을 지님

$$D_{KL}(p||q) \geq 0$$

  - 두 분포가 같을 때 값이 0, 분포가 다를 수록 값이 커짐
  - 계산 순서가 바뀌면 값이 변할 수 있음

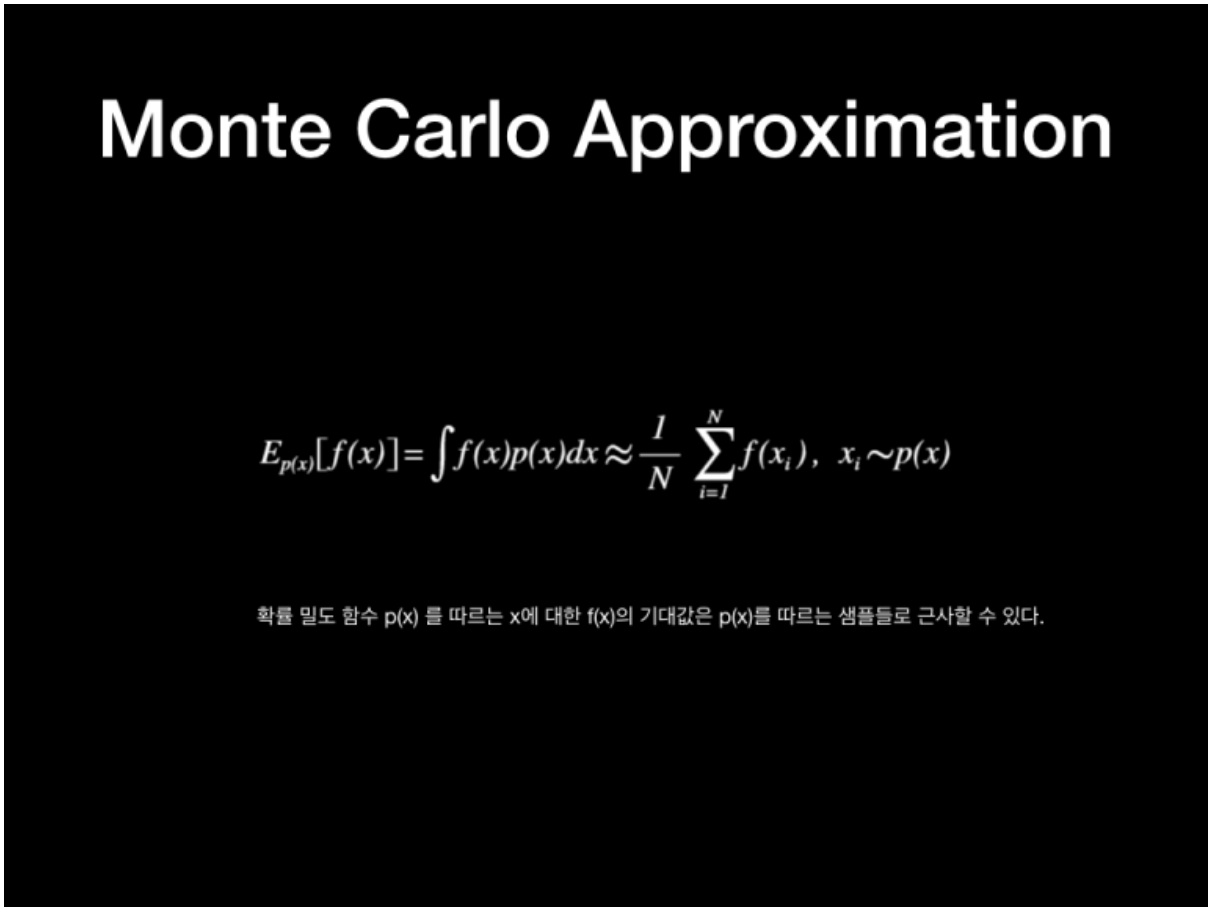
$$D_{KL}(p||q) \neq D_{KL}(q||p)$$

$$\int p(x) \log p(x)\,dx - \int p(x) \log q(x)\,dx \neq \int q(x) \log q(x)\,dx - \int q(x) \log p(x)\,dx$$



만약 다름을 비교할 두 확률분포가 정규분포를 따를 경우에 KLDivergence의 식은 위와같이 전개되어 간략하게 표현이 가능해진다.

Monte Carlo Approximation



무작위 알고리즘은 크게 2가지 종류가 있다.

하나는 MC(Monte Carlo) 알고리즘이고,

다른 하나는 Las Vegas 알고리즘이다.

라스베가스 알고리즘은 항상 정확한 답을 주거나, 아니면 아예 답을 주지 못한다

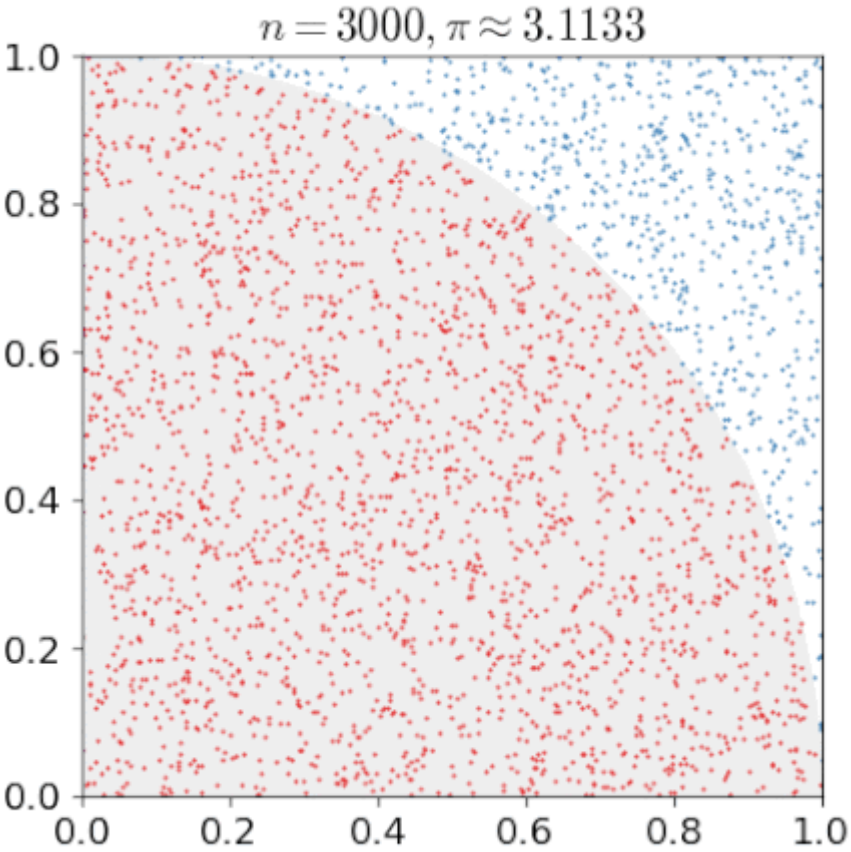
반면 MC는 항상 정확한 답을 주지는 않지만, 어느 정도의 자원 투입이 보장되면, 상당한 수준으로 ‘heuristic’하게 답을 근사할 수 있다.

ML기반의 문제 해결은, 대부분 정답을 근사하는 경우가 많다. 상기했듯이,

$$q_\phi(z) \simeq f_\theta(z)$$

와 같이, 사후확률을 구하는 것이 어려우므로, 근사하는 것과 마찬가지로 VAE 기저에 깔린 Randomization Algorithm의 경우는 MC 방법을 선택한다는 것을 알 수 있다.

기본적으로 몬테카를로 방법은 무작위로 난수를 생성하고, 특정 연산에 입력값으로 넣어 결과를 추출한다. 이런 과정을 여러 번 반복하면 연산의 파라미터값을 추정할 수 있다.



이렇게, 원주율 pi를 구함에 있어서도 무작위 난수 생성 후 점점 근사시키는 것을 알 수 있다.

이 방법에서 이 방법이 얼마나 정확한지를 평가할 수도 있는데, DL에서의 loss function, Evidence of Lower BOund 등을 설정하면 모델의 오차를 평가할 수도 있다.

MC 알고리즘은,

하기의 regularization error 계산에 있어서

$$\mathbb{E}_{q_\phi(z|x_i)}[log(\, p_\theta(x_i|z))] = \int log(\, p_\theta(x_i|z)\,)\, q_\phi(z|x_i)dz \approx \frac{1}{L}\sum_{z^{i,l}} log\left(p_\theta(x_i|z^{i,l})\right) \approx log\left(p_\theta(x_i|z^i)\right)$$

Monte-carlo technique

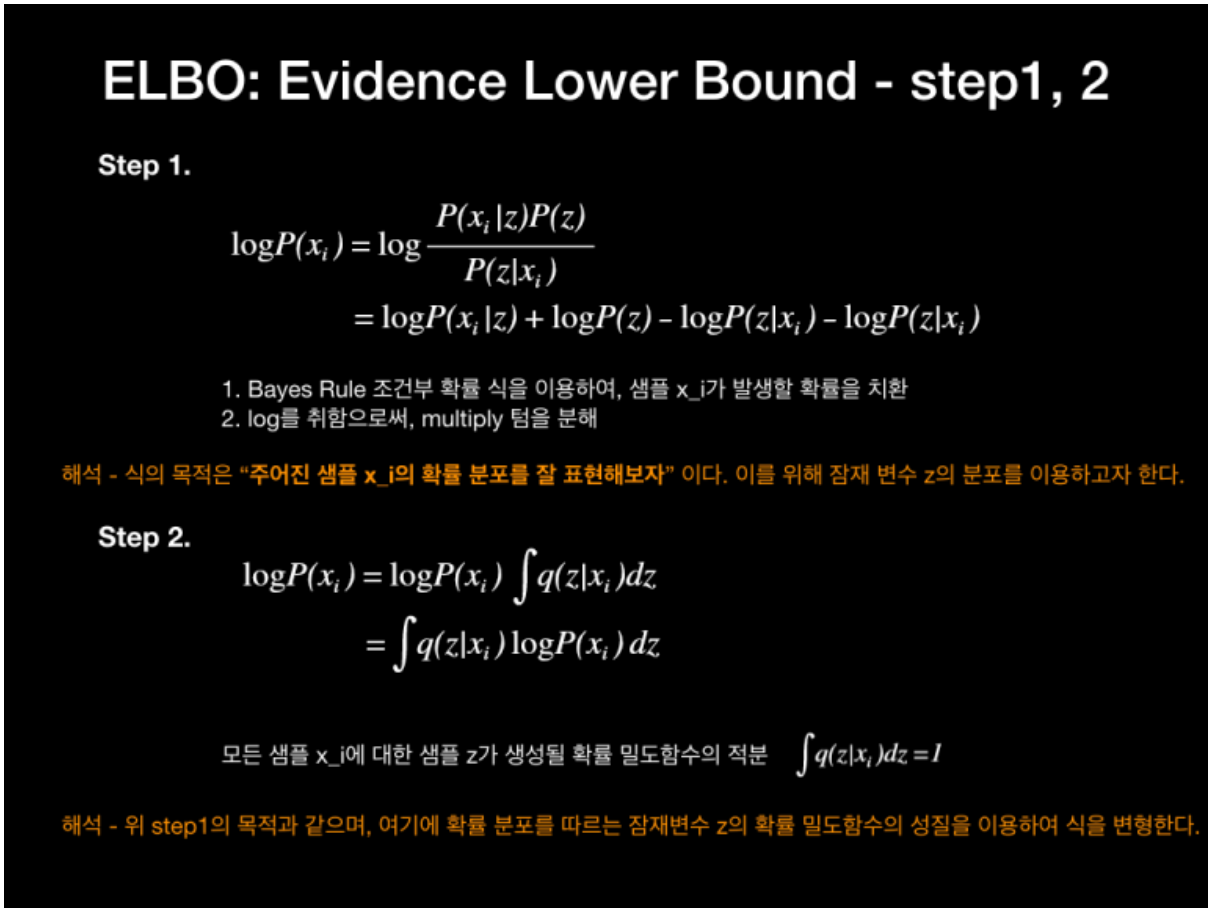
L=1

를 근사하는데 쓰인다.

즉,

“확률 밀도 함수 p(x)를 따르는 x에 대한 f(x)의 기대값은 p(x)를 따르는 샘플들로 근사할 수 있다”
라고 할 수 있다.

ELBO(Evidence Lower BOund)



그래서

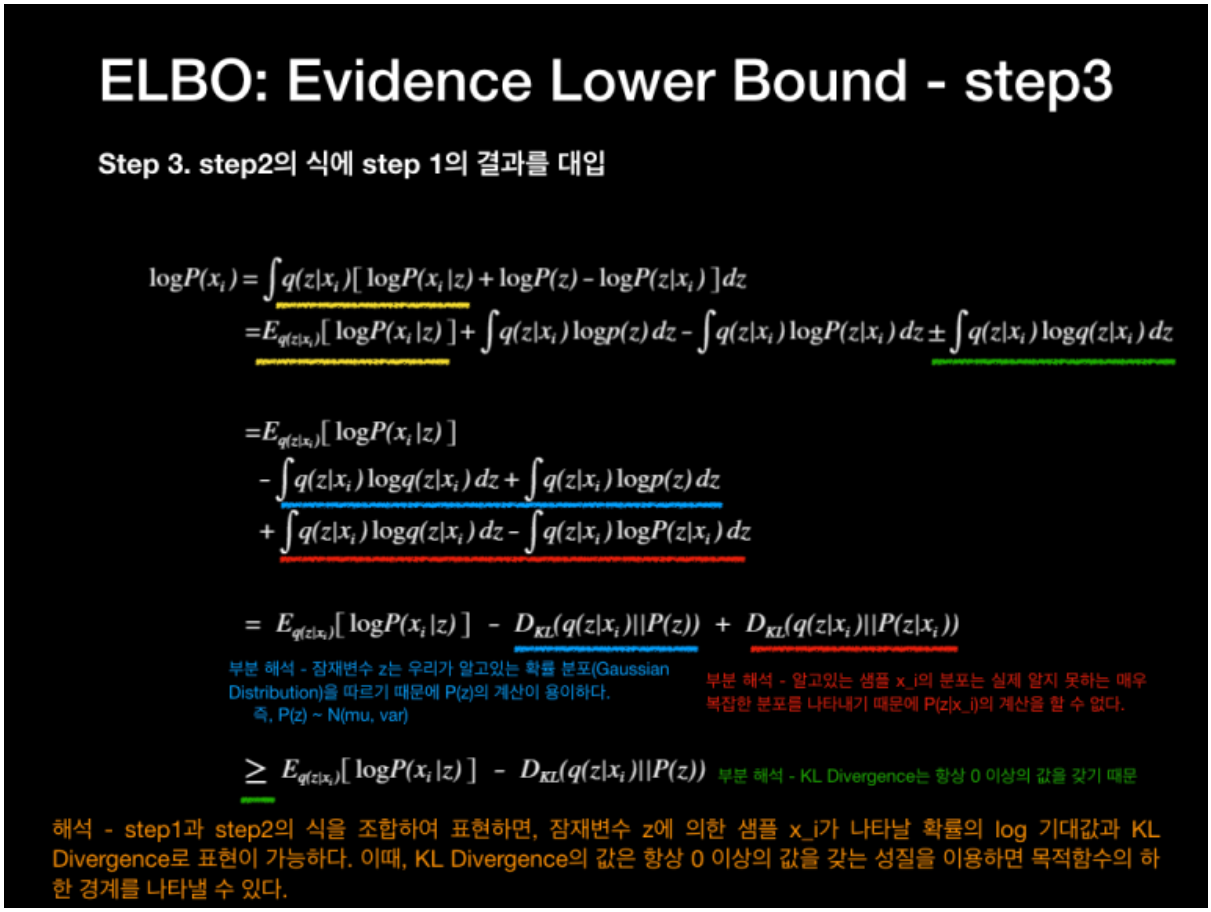
Step1) 주어진 샘플 X의 확률 분포를 잘 표현해보자
라는 것을 식으로 표현하면 다음과 같다.

P(X)는 베이즈 정리(Bayes Theorem)에 의해 치환될 수 있고, 이때 우리는 (알고있는) Latent Variables z의 분포를 이용한다.

여기서 ‘알고있는’이라는 표현이 쓰이는 이유는, VAE의 focus는 output을 latent variable로부터 ‘생성’하는데 포커스가 맞추어져 있기 때문이다.

그리고 확률 P(X)는 잠재변수 z의 확률 밀도함수의 성질을 통해 Step2와 같이 나타낼 수 있다.

P(x)의 식에 다시 자기 자신을 포함하는 재귀 형태로 식이 표현된다.



Step1의 식을 Step2의 식에 대입하여 식을 전개한다.

→ 노란색 부분은: Monte Carlo Approximation에 의해 기대값으로 근사하여 표현 가능

→ 초록색 밑줄과 같은 term을 더하고 빼서 식의 등식이 유지될 수 있게 함

이 과정에서,

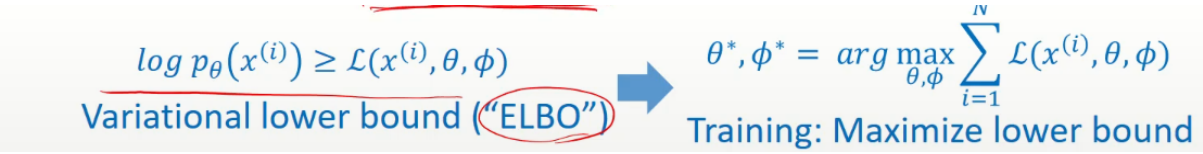
KL Divergence의 식이 위 식 전개과정에서 생겨나는 것을 볼 수 있다.

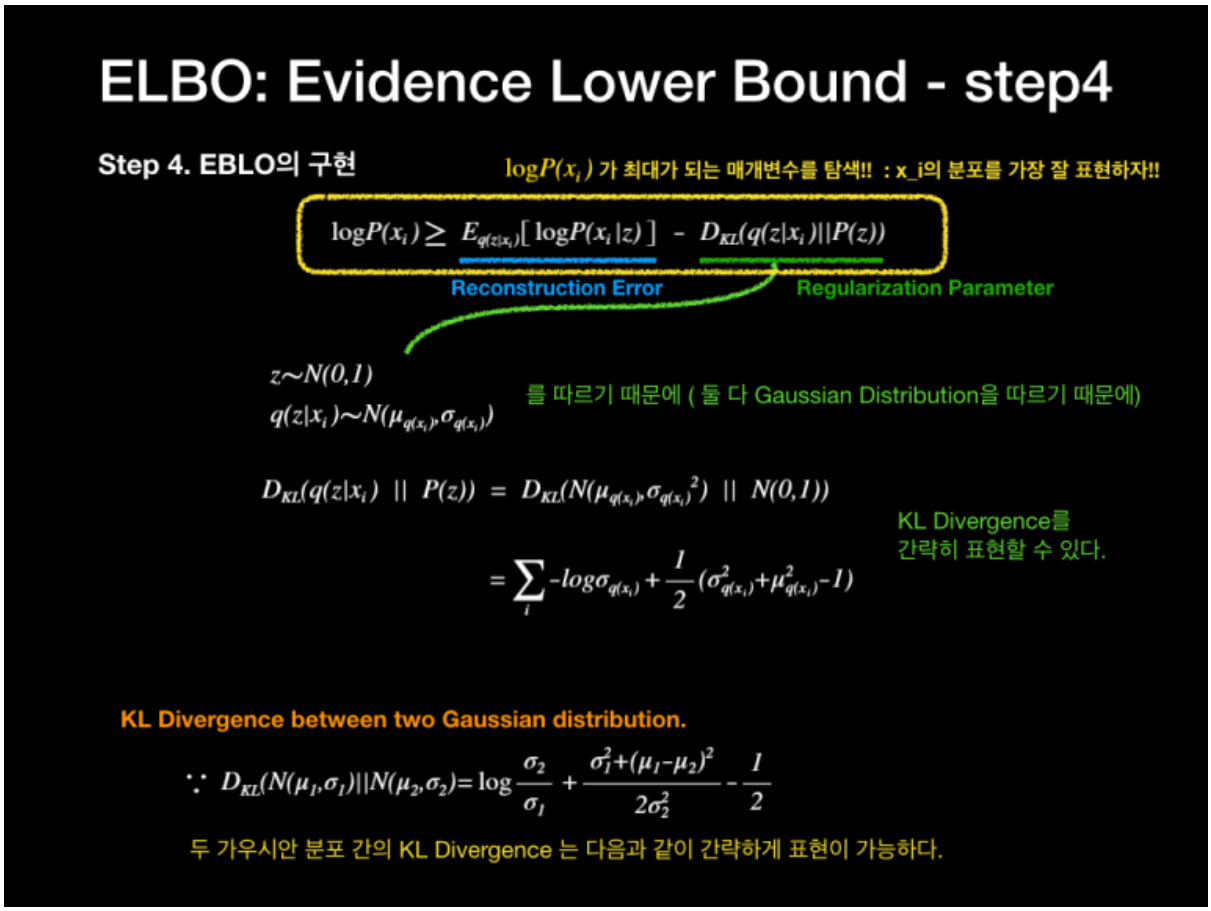
이 KLD를 어떻게 처리할 지에 대한 논의가 필요하다.

→ 파란색 밑줄 부분은 잠재변수 z가 우리가 알고있는 확률 분포(정규분포)를 따르기 때문에 KLD의 계산이 용이하다.

→ 빨간색 밑줄 부분은 가지고 있는 샘플 X의 복잡한 분포를 우리는 실제로 알고있지 못하기 때문에 P(z|x_i) 를 계산이 불가하다. 다만, KLD는 항상 0 이상의 값을 갖는다는 특성이 존재한다.

그래서 빨간색 밑줄 부분을 제외한 나머지 부분은 항상 본 식보다 작거나 같다(Evidence Lower BOund) 라는 부등호로 식이 만들어진다.





결국 목적은, 가지고 있는 데이터 X의 분포를 잘 표현하는 것이다.

결국, 분포를 잘 표현하고자 하는 목적에 부합하기 위해서는 log P(x)가 최대가 되는 방향으로 모델을 학습하면 되는 것이다.

KLD 부분의 식의 구현은, q(z|x_i)는 주어진 샘플 x_i에 대해 Encoder를 통과하여 얻어진 mu와 sigma를 따르는 정규분포가 될 것이며,

P(z)는 z~N(0,1)를 만족한다고 가정할 때, 모두 정규분포이므로 간단하게 표현이 가능하다.

여기서 노란색 부분을 뜯어서 살펴보자.

문제를 바꾸어서,

기존 목적함수의 argmax parameter를 구하는 문제에서

→ 목적함수에 -를 곱하여 modified 목적함수의 argmin parameter를 구하는 문제로서 바꾸어 생각해보자.

$$arg\min_{\theta,\phi} \sum_i \underbrace{-\mathbb{E}_{q_\phi(z|x_i)}\left[log\left(p(x_i|g_\theta(z))\right)\right]}_{\text{Reconstruction Error}} \underbrace{+KL(q_\phi(z|x_i)||p(z))}_{\text{Regularization}}$$

Reconstruction Error

- 샘플링용 함수에 대한 negative log likelihood
- 결국 입력 데이터가 encoder를 통해 정규분포를 따르는 어떤 공간에 표현되고, 이로부터 다시 decoder를 통과했을 때 x가 되게끔 하는 것을 목적으로 함\
- x_i에 대한 복원 오차(AutoEncoder 관점)

Regularization

- 현재 샘플링용 함수에 대한 추가 조건
- 샘플링의 용의성/생성데이터에 대한 통제성을 위한 조건을 prior에 부여하고, 이와 유사해야 한다는 조건을 부여