VAE: Auto-Encoding Variational Bayes

Diederik P. Kingma, Max Welling

VGAE: Variational Graph Auto-Encoders

Thomas N. Kipf. Max Welling

박진혁

2023.08.01

CONTENTS

Introduction

VAE = Variational Inference + Auto Encoder

Generative Model이다

Introduction

Inference distribution – 기존의 분포 추정

1. MLE

MLE는 관측치 X가 주어졌을 때, 관측치 x가 등장할 확률의 곱인 likelihood p(x)를 사용합니다.
Likelihood를 parameter에 관해 최대화화는 parameter를 구한다

MLE는 주어진 데이터를 잘 설명하지만, 데이터 분포에 대한 가설을 활용하지 않는다

(주사위를 던져서 1이 3번 4가 1번 나왔다면 다른 확률은 없는가?)

$$p(x; \theta_1, \theta_2, \dots, \theta_6) = \prod_{i=1}^{6} \theta_i^{\mathbf{I}_i(x)}$$

2. MAP

MAP는 데이터 분포에 대한 가설을 활용한다. 이를 prior p(θ)라고 한다.

이후 관측치 x를 고려하여 이를 업데이트한다. 즉, prior가 likelihood의 가중치로 사용된다.

$$p(\theta|X) = \frac{p(X|\theta)p(\theta)}{p(X)} \qquad \theta_{MAP} = \arg\max \prod_i p(x_i|\theta)p(\theta)$$

https://modulabs.co.kr/blog/variational-inference-intro/

Introduction

Variational Inference

Variational Inference란 사후(posterior) 분포를 p(z|x)를 q(z)로 근사하는 것을 말한다

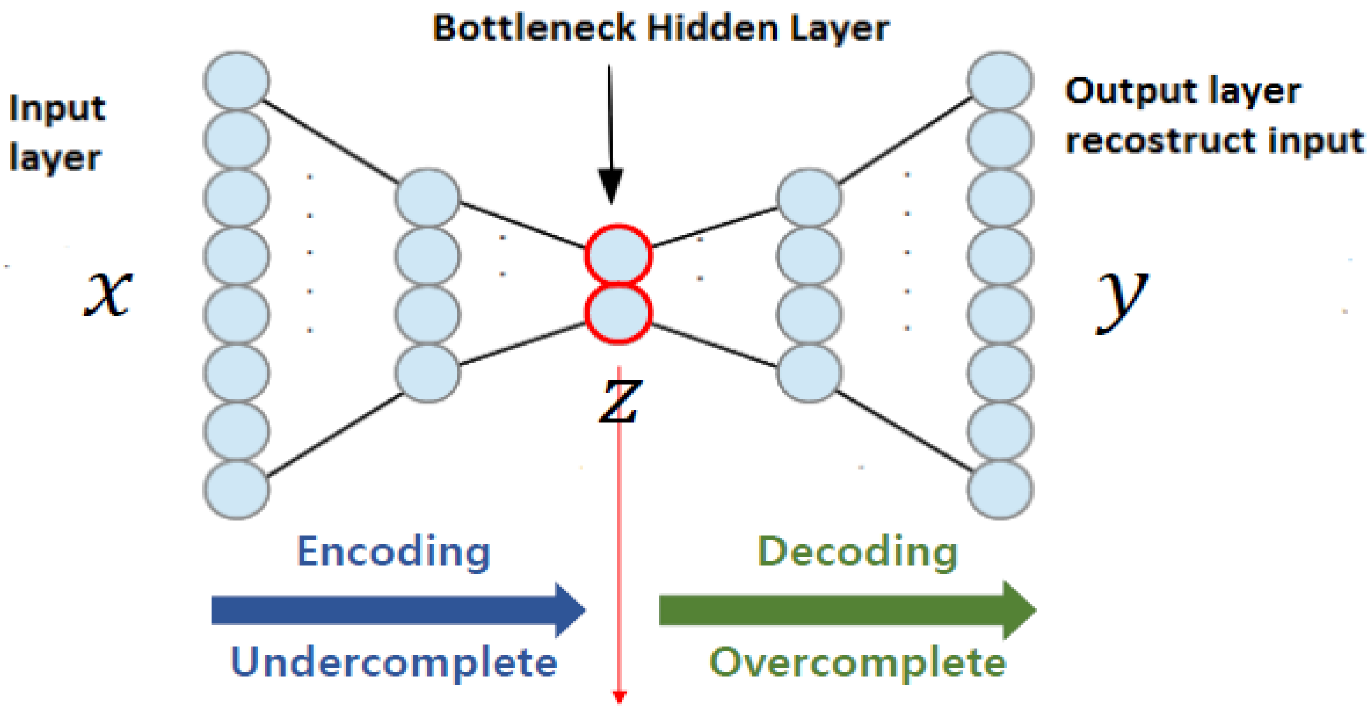
1. Marginal probability: 사후확률의 분모인 p(x)를 계산하기 어려운 경우
2. Likelihood 계산 시
3. Prior p(z) 모델링시

사용하는 경우가 많다

https://velog.io/@chulhongsung/VI

Introduction

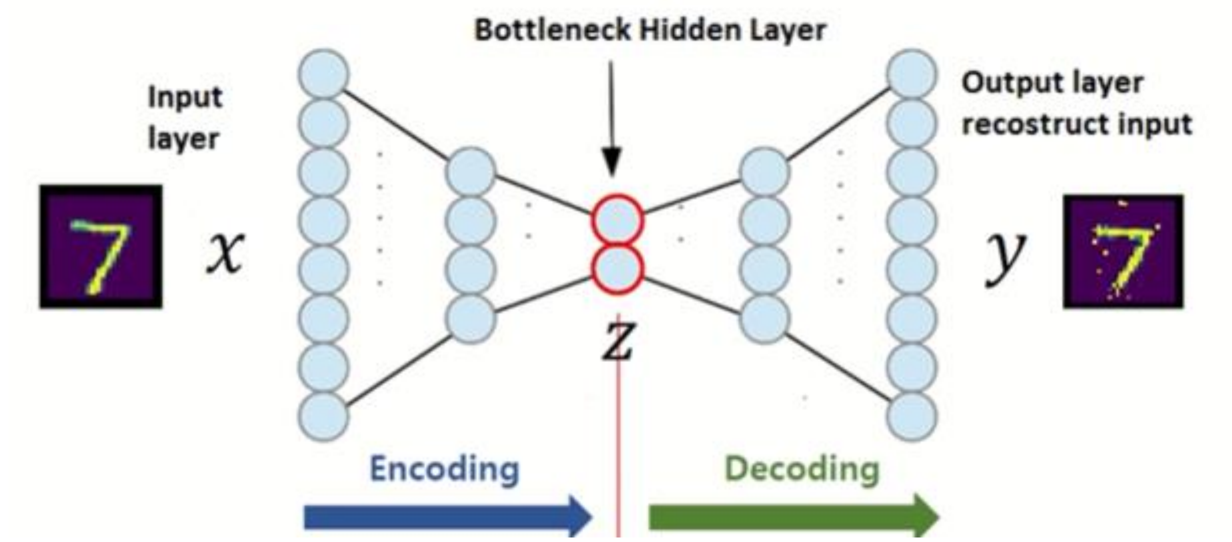
Auto-Encoder



- Code
- Latent Variable
- Feature
- Hidden representation

Background

Auto-Encoder



Loss function은 확률분포에 따라 MSE 혹은 Cross entropy을 사용한다

Auto Encoder란 input과 ouput이 동일한 네트워크를 말한다

구조는 Encoder, Decoder로 구성되어있다.

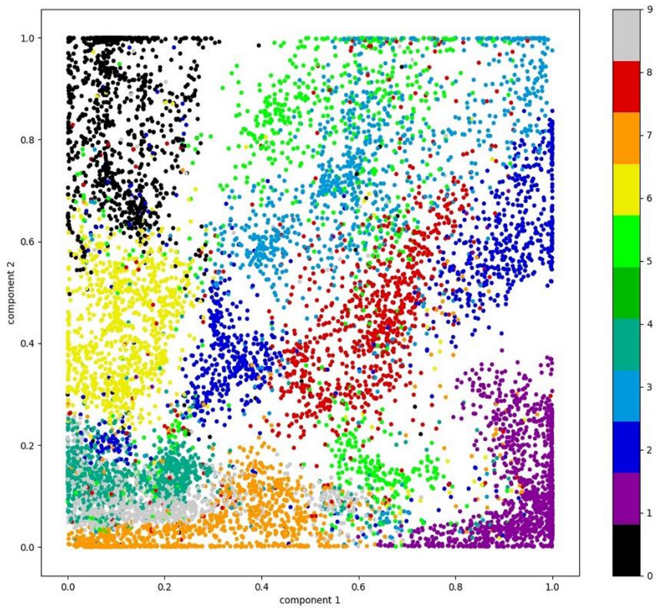
encoder는 학습데이터를 주요한 latent vector(or space)로 표현하고, decoder는 학습데이터를 잘 만들어내는 방법이다

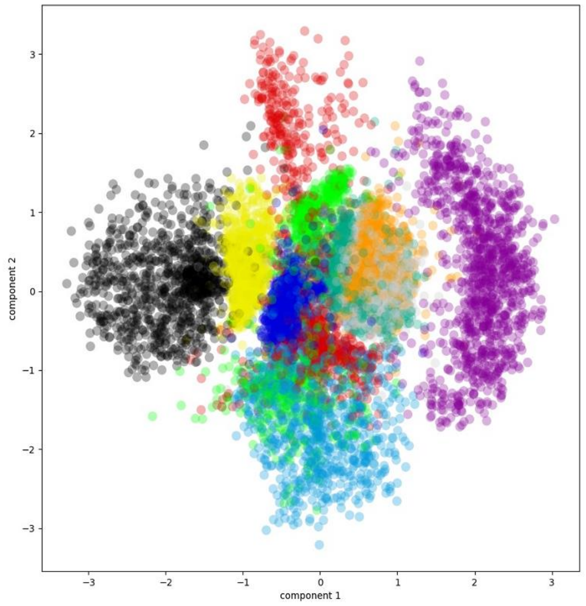
Auto Encoder가 주목받았던 이유는 Unsupservised Learning문제를 Supervised Learning으로 바꾸어서 해결했기에 주목받았다(나 자신, x가 label로 작용되기에)

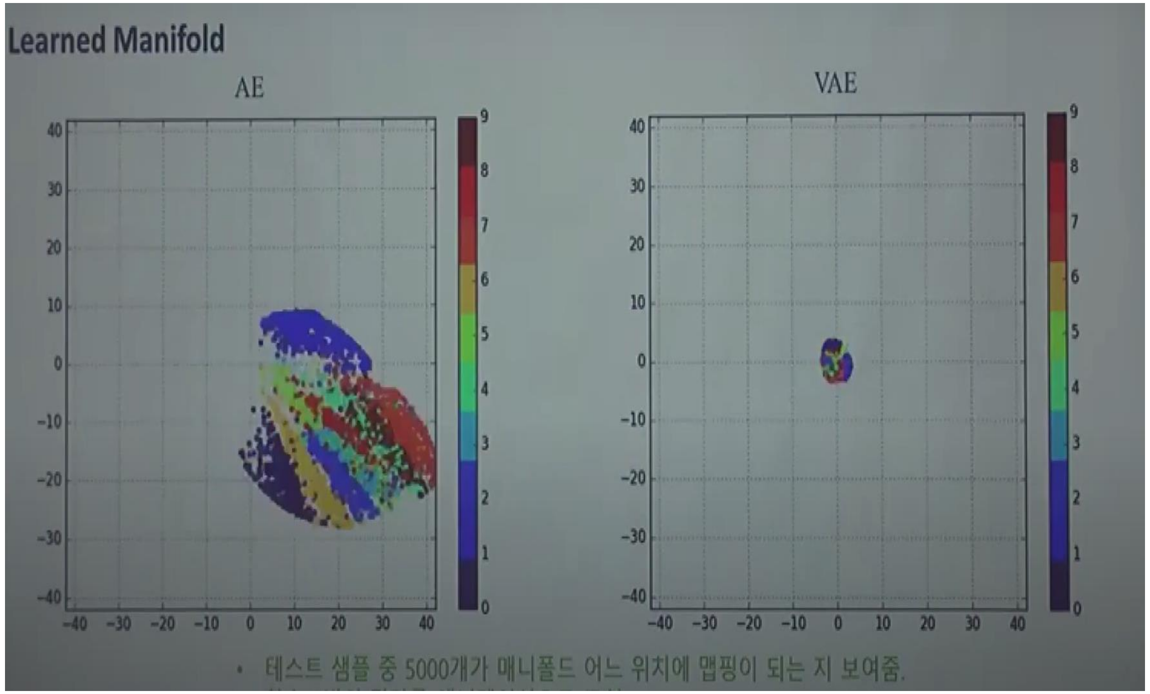
즉, z = h(x), y= g(z) = g(h(x))이다. 이때 Loss는 x,y의 loss function으로 구성된다.

Introduction

Why VAE?







Manifold 관점에서 VAE와 AE의 차이가 있음(VAE는 AE보다 훨씬 안정적이고 뭉쳐져 있다)

1. 안정적이다 == AE의 경우 에폭마다 계속 subspace 위치가 바뀜, z sampling 관점에서 중요
2. 뭉쳐져 있다 == AE는 최대한 데이터가 구별되게 만듬(VAE 아키텍처는 이미지 표현이 서로 가까워지도록 하여 데이터 포인트 사이의 알려지지 않은 영역 간격을 크게 줄인다)

https://www.youtube.com/watch?v=rNh2CrTFpm4

Background

Entropy, Cross Entropy, KL divergence

$$H(x) = -\sum_{i=1}^{n} p(x_i) log p(x_i)$$

$$H_p(q) = -\sum_{i=1}^{n} q(x_i) log p(x_i)$$

$$D_{KL}(q||p) = -\sum_{c=1}^{C} q(y_c)[log(p(y_c)) - log(q(y_c))] = H_p(q) - H(q)$$

$$D_{\mathrm{KL}}(p \parallel q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

Entropy:
정보를 표현하는데 있어서 필요한 최소 자원량, 확률이 적은 애를 길게 작성하고 확률이 큰 아이를 짧게 작성해라

Cross Entropy:
원래의 cross entropy는 예측 모형은 실제 분포인 q 를 모르고, 모델링을 하여 q 분포를 예측하고자 하는 것이다

KB divergence: “KL Divergence” 라고 주로 부르는 서로 다른 두 분포의 차이 (dissimilarity) 를 측정하는데 쓰이는 measure 이다. 이를 entropy와 cross entropy 개념에 대입하면 두 entropy 차이로 계산된다

Background

손실함수의 정의 기준

Loss function은 최적해가 관측데이터를 잘 설명할 수 있는 함수의 파라미터 값이 되도록 정의 되어야 한다

해당 정의 기준은 크게

1. 오차 최소화(error minimization): MSE, MAE
2. 최대 우도 추정(MLE) 2가지 관점이 존재한다

최대 우도 추정은 모델이 추정하는 관측 데이터의 확률이 최대화 되도록 정의하는 방법이다.

우도 식은 아래와 같으며 아래 식이 최대가 될 때의 모수를 찾는 것이 최대우도추정이다.

$$P(x|\theta) = \prod_{k=1}^{n} P(x_k|\theta)$$

우리는 대다수 loss function을 minimization 시키기에 이 식을 negative log likelihood 식으로 변형하여 사용한다

Background

MLE 관점에서 Loss function의 비교

Log-Likelihood for Neural Nets

- Estimating a conditional probability $P(Y|X)$
- Parametrize it by $P(Y|X) = P(Y|\omega = f_\theta(X))$
- Loss = $-\log P(Y|X)$
- E.g. Gaussian *Y*, $\omega = (\mu, \sigma)$
  typically only μ is the network output, depends on *X*
  Equivalent to MSE criterion:
  Loss = $-\log P(Y|X) = \log \sigma + ||f_\theta(X) - Y||^2/\sigma^2$
- E.g. Multinoulli *Y* for classification,
  $\omega_i = P(Y = i|x) = f_{\theta,i}(X) = \mathrm{softmax}_i(a(X))$
  Loss = $-\log \omega_Y = -\log f_{\theta,Y}(X)$

각 모델이 추정하는 분포에 따라 loss function이 다르게 나온다.

1. Gaussian Distribution -> MSE

2. Bernoulli distribution -> Cross entropy

Yoshua Bengio's의 slide

Background

MLE 관점에서 Gaussian & Bernoulli distribution

Univariate cases

$$-\log(p(y_i|f_\theta(x_i)))$$

Gaussian distribution

$$f_\theta(x_i) = \mu_i, \sigma_i = 1$$

$$p(y_i|\mu_i, \sigma_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(y_i - \mu_i)^2}{2\sigma_i^2}\right)$$

$$\log(p(y_i|\mu_i, \sigma_i)) = \log\frac{1}{\sqrt{2\pi}\sigma_i} - \frac{(y_i - \mu_i)^2}{2\sigma_i^2}$$

$$-\log(p(y_i|\mu_i)) = -\log\frac{1}{\sqrt{2\pi}} + \frac{(y_i - \mu_i)^2}{2}$$

$$-\log(p(y_i|\mu_i)) \propto \frac{(y_i - \mu_i)^2}{2} = \frac{(y_i - f_\theta(x_i))^2}{2}$$

Mean Squared Error

Bernoulli distribution

$$f_\theta(x_i) = p_i$$

$$p(y_i|p_i) = p_i^{y_i}(1 - p_i)^{1-y_i}$$

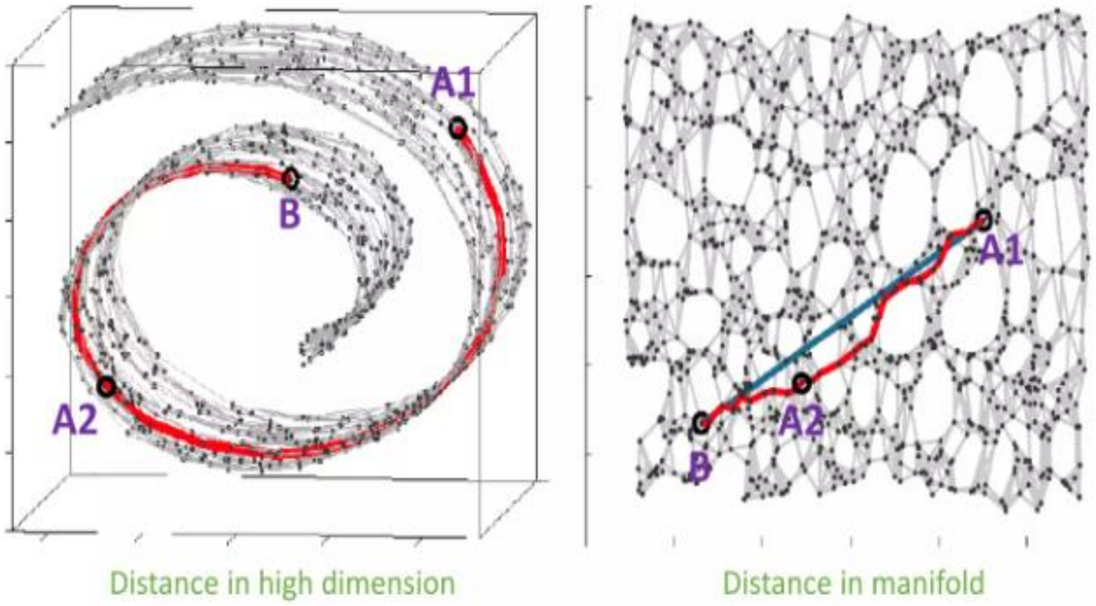
$$\log(p(y_i|p_i)) = y_i \log p_i + (1 - y_i)\log(1 - p_i)$$

$$-\log(p(y_i|p_i)) = -[y_i \log p_i + (1 - y_i)\log(1 - p_i)]$$

Cross-entropy

Background

Manifold Learning & Dimension Reduction



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

Manifold learning은 쉽게 말해 고차원 데이터에서 이 데이터들을 잘 포함하는 subspace(manifold)가 존재할 것이다! 라고 예측을 하고 해당 subspace를 찾는 것이다.

이렇게 하면 아래의 4가지 문제를 해결할 수 있다.

1. data compression
2. data visualization
3. curse of dimensionality
4. discovering imporatant features

: resonable distance metirc

Background

Manifold Learning & Dimension Reduction

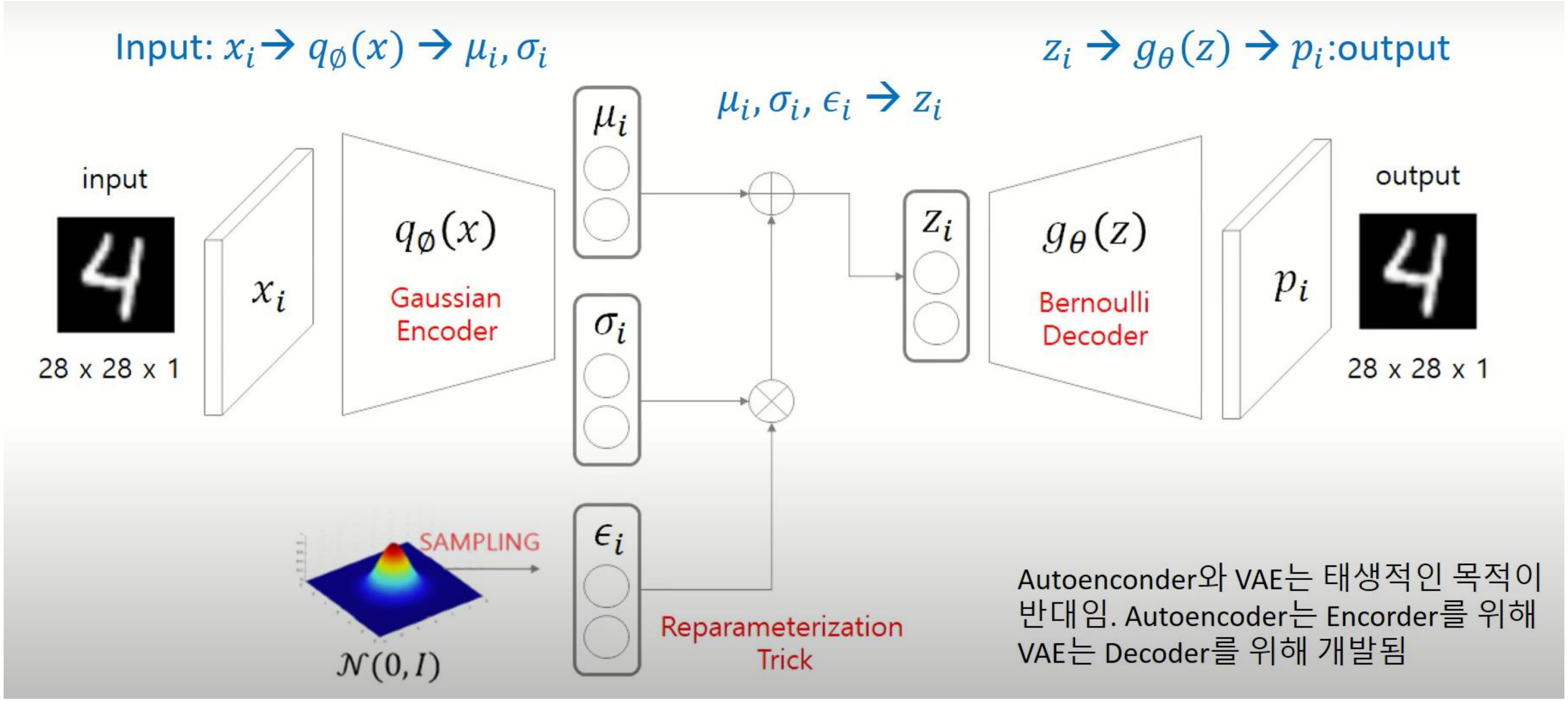
Dimensionality Reduction

Linear
- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- etc..

Non-Linear
- Autoencoders (AE)
- t-distributed stochastic neighbor embedding (t-SNE)
- Isomap
- Locally-linear embedding (LLE)
- etc..

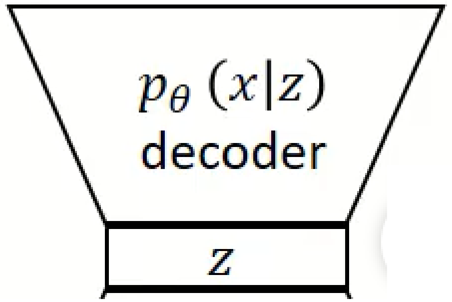
Model description

VAE(Variational Auto Encoder)



Model description

Problem 1: Intractability



Why intractable?

Learn model parameters to maximize
likelihood of training data

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Z를 알 수 없다
Unfortunately, a lot of this process is hidden from our view: the true parameters as well as the values of the latent variables z(i) are unknown to us.

p(z)p(x|z) = p(x,z), p(x,z) is joint probability distribution -> marginal distribution으로 변경

x: input & output
Z: latent vector
P(x|z): z가 주어졌을 때 x가 생성되는 확률
P(x|z)를 간단한 distributio으로 해도 되는 이유:
DNN이라 문제없다

Model description

Problem 2: A large dataset & large cost

Learn model parameters to maximize
likelihood of training data

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

x: input & output
Z: latent vector
P(x|z): z가 주어졌을 때 x가 생성되는 확률

we have so much data that batch optimization
is too costly; we would like
to make parameter updates using small
minibatches or even single datapoints.
Sampling based
solutions, e.g. Monte Carlo EM, would in
general be too slow, since it involves a
typically expensive sampling loop per
datapoint.

-> Monte Carlo EM과 같은 sampling 기반은
cost가 너무 크다, 이를 해결하고 싶다

+ 일반적인 prior(p(x)에서) uniform sampling
으로 해결할 수 없음

Model description

So we are interested in

CASE1:

Parameter θ에 대한 Efficient approximate Maximum likelihood 혹은 MAP 추정
-> 매개 변수 자체에 관심이 있을 수도 있고, real data를 닮은 인공 데이터를 생성하도록 도와줄 수 있음

CASE2:
X가 주어였을 때 z에 대한 효율적인 approximate posterior inference -> 실제 cost를 줄이고 싶다

CASE 3:
x의 prior가 요구되는 모든 종류의 inference task 수행에 관심이 있다

Model description

So to Solve this…

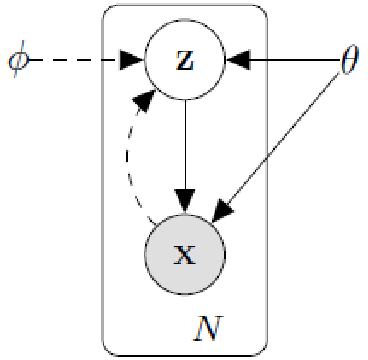
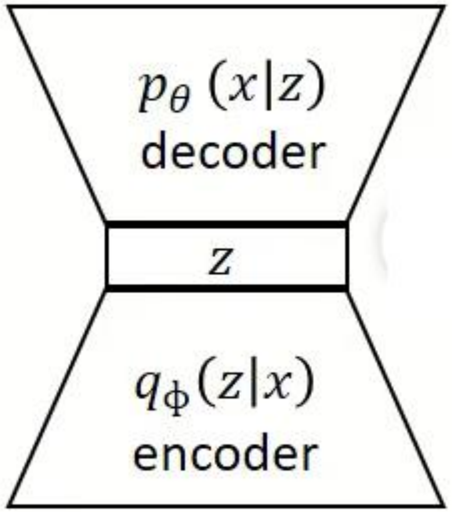


Figure 1: The type of directed graphical model under consideration. Solid lines denote the generative model $p_{\boldsymbol{\theta}}(\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, dashed lines denote the variational approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to the intractable posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. The variational parameters $\boldsymbol{\phi}$ are learned jointly with the generative model parameters $\boldsymbol{\theta}$.



이를 해결하기 위해 decoer network modeling을 위해 recognition model $q_\phi(z|x)$를 추가하자. 이는 true x's distribution p(z|x)를 근사한 것이다(이상적인 sampling 함수가 필요하다)

이 θ, Φ는 동시에 학습하는 방법을 도입하였고, z에 대해서도 recognition model $q_\phi(z|x)$를 통해서 x가 주어졌을 때 X가 생성될 수 있는 지점인 code z의 가능한 값들에 관한 분포를 만들어 낸다.
$p_\theta(x|z)$는 우리는 확률적 decoder로 지칭하여 code z가 주어졌을 때 가능한 x의 분포를 만들어낸다.

Model description

ELBO term & maximum likelihood

The marginal likelihood is composed of a sum over the marginal likelihoods of individual datapoints $\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}) = \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$, which can each be rewritten as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\boldsymbol{\theta}, \phi; \mathbf{x}^{(i)}) \tag{1}$$

The first RHS term is the KL divergence of the approximate from the true posterior. Since this KL-divergence is non-negative, the second RHS term $\mathcal{L}(\boldsymbol{\theta}, \phi; \mathbf{x}^{(i)})$ is called the (variational) *lower bound* on the marginal likelihood of datapoint i, and can be written as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\boldsymbol{\theta}, \phi; \mathbf{x}^{(i)}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[-\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right] \tag{2}$$

which can also be written as:

$$\mathcal{L}(\boldsymbol{\theta}, \phi; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right] \tag{3}$$

Model description

ELBO term & maximum likelihood

Elbo 유도식

$$\log p_\theta(x^{(i)}) = E_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right]$$

$$= E_z\left[\log \frac{p_\theta(x^{(i)}|z)\,p_\theta(z)}{p_\theta(z|x^{(i)})}\right] \quad \left(\because P(A|B) = \frac{P(A\cap B)}{P(B)} = \frac{P(A)P(B|A)}{P(B)}\right)$$

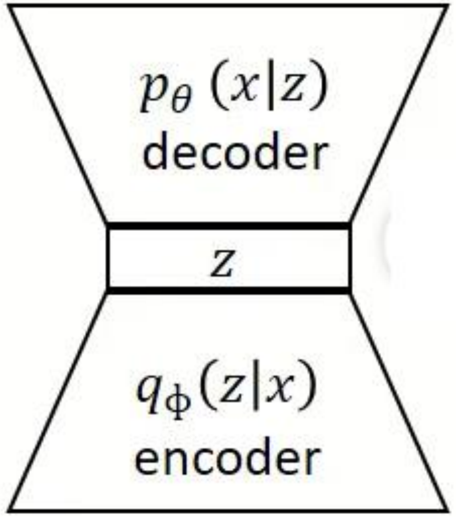
$$= E_z\left[\log \frac{p_\theta(x^{(i)}|z)\,p_\theta(z)}{p_\theta(z|x^{(i)})} \times \frac{q_\phi(z|x^{(i)})}{q_\phi(z|x^{(i)})}\right]$$

$$= E_z\left[\log p_\theta(x^{(i)}|z)\right] - E_z\left[\log \frac{q_\phi(z|x^{(i)})}{p_\theta(z)}\right] + E_z\left[\log \frac{q_\phi(z|x^{(i)})}{p_\theta(z|x^{(i)})}\right]$$

$$= E_z\left[\log p_\theta(x^{(i)}|z)\right] - D_{KL}\left(q_\phi(z|x^{(i)}) \,\|\, p_\theta(z)\right) + D_{KL}\left(q_\phi(z|x^{(i)}) \,\|\, p_\theta(z|x^{(i)})\right)$$

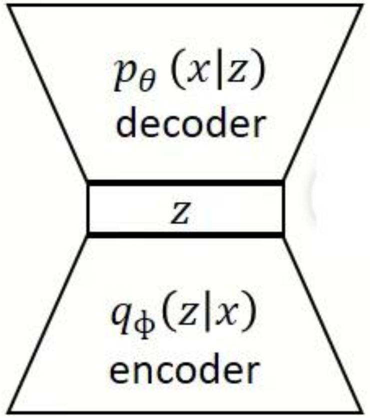
$$\left(\because E_{z\sim q_\phi(z|x^{(i)})} \log \frac{q_\phi(z|x^{(i)})}{p_\theta(z)} = \underbrace{\int_z \log \frac{q_\phi(z|x^{(i)})}{p_\theta(z)}\, q_\phi(z|x^{(i)})\, dz}_{\text{KL term}}\right)$$

$$KL(p\,\|\,q) = \begin{cases} \sum_i p_i \log \frac{p_i}{q_i} \text{ 또는 } -\sum_i p_i \log \frac{q_i}{p_i} & \text{(이산형)} \\ \int p(x) \log \frac{p(x)}{q(x)} dx \text{ 또는 } -\int p(x) \log \frac{q(x)}{p(x)} dx & \text{(연속형)} \end{cases}$$



Model description

ELBO term



Elbo 유도식

$$\log p_\theta(x^{(i)}) = E_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right]$$

Maximize the likelihood of x

$$= E_z\left[\log \frac{p_\theta(x^{(i)}|z)\,p_\theta(z)}{p_\theta(z|x^{(i)})}\right] \quad \left(\because P(A|B)=\frac{P(A\cap B)}{P(B)} = \frac{P(A)P(B|A)}{P(B)}\right)$$

$$= E_z\left[\log \frac{p_\theta(x^{(i)}|z)\,p_\theta(z)}{p_\theta(z|x^{(i)})} \times \frac{q_\phi(z|x^{(i)})}{q_\phi(z|x^{(i)})}\right]$$

$$= E_z\left[\log p_\theta(x^{(i)}|z)\right] - E_z\left[\log \frac{q_\phi(z|x^{(i)})}{p_\theta(z)}\right] + E_z\left[\log \frac{q_\phi(z|x^{(i)})}{p_\theta(z|x^{(i)})}\right]$$

$$= E_z\left[\log p_\theta(x^{(i)}|z)\right] - D_{KL}\left(q_\phi(z|x^{(i)}) \,\|\, p_\theta(z)\right) + D_{KL}\left(q_\phi(z|x^{(i)}) \,\|\, p_\theta(z|x^{(i)})\right)$$

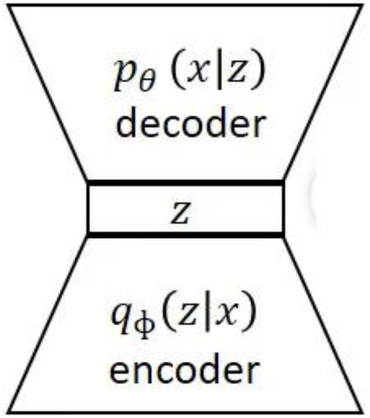
Decoder give P(x|z) by Sampling z

$$\left(\because E_{z\sim q_\phi(z|x^{(i)})} \log \frac{q_\phi(z|x^{(i)})}{p_\theta(z)} = \underbrace{\int_z \log \frac{q_\phi(z|x^{(i)})}{p_\theta(z)}\, q_\phi(z|x^{(i)})\, dz}_{\text{KL term}}\right)$$

KL term – z prior vs encoder part

Model description

ELBO



$$\log p_\theta(x^{(i)})$$

$$= \boxed{\mathbf{E}_z[\log p_\theta(x^{(i)}|z)] - D_{kL}\left(q_\phi(z|x^{(i)})\,||p_\theta(z)\right)} + D_{kL}\left(q_\phi(z|x^{(i)})\,||p_\theta(z|x^{(i)})\right)$$

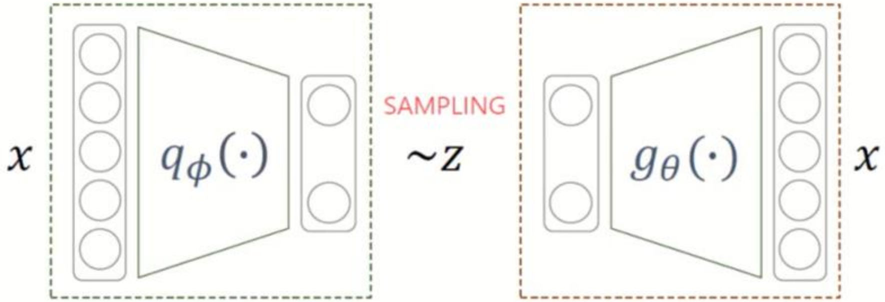
ELBO term

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\boldsymbol{\theta}, \phi; \mathbf{x}^{(i)})$$

Intractable: p(z|x)를 모른다

Model description

Loss function



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

Reconstruction Error

- 현재 샘플링용 함수에 대한 negative log likelihood
- x_i에 대한 복원 오차 (Autoencoder 관점)

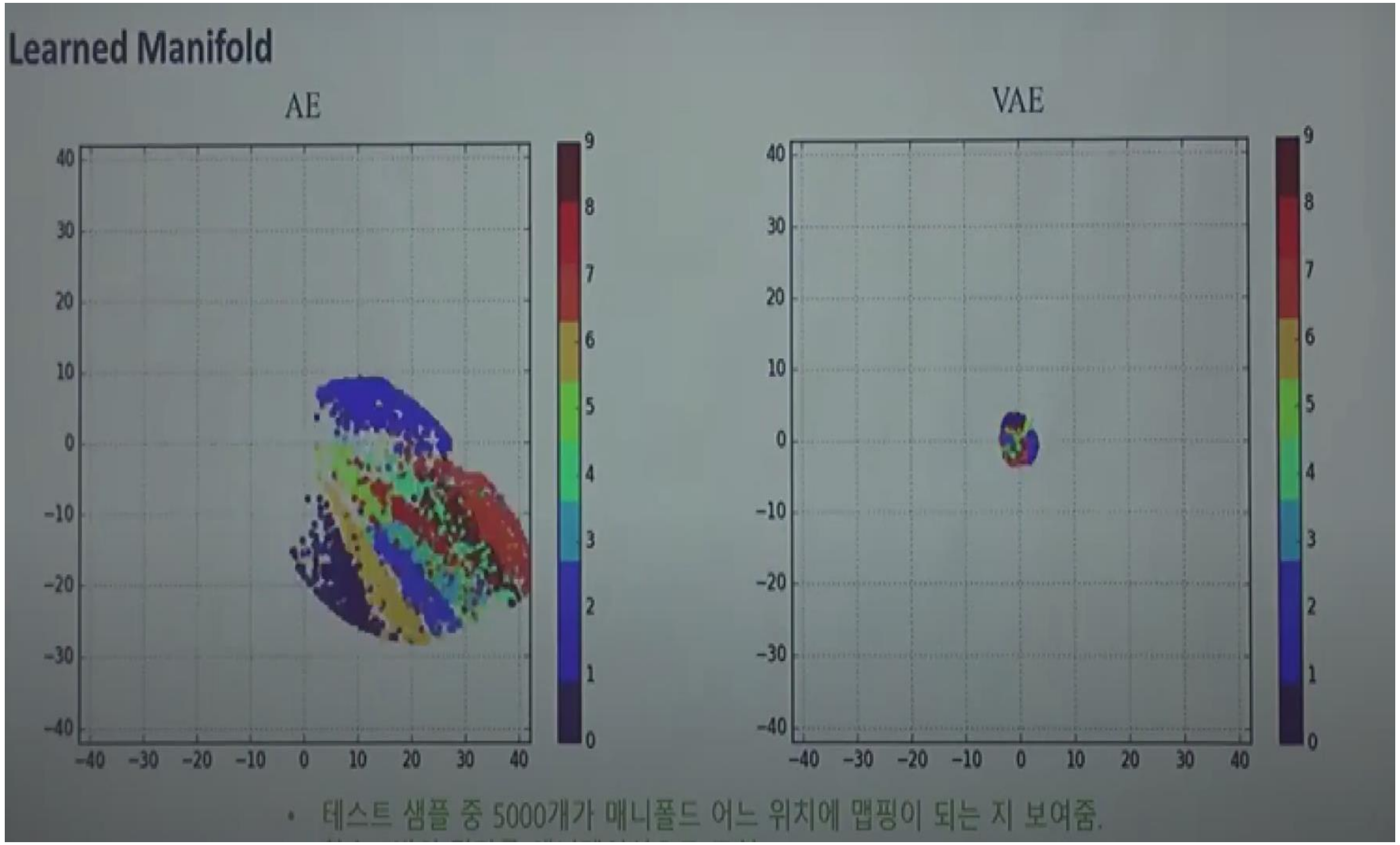
Regularization

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

참고: $p(x|g_\theta(z)) = p_\theta(x|z)$

Model description

Regularization part case 2:



Z에 관한 이상적인 샘플링 함수와

생성하는 부분이 같이 최적화(p(z)도 다루기 쉬운 간단한 분포)되기에

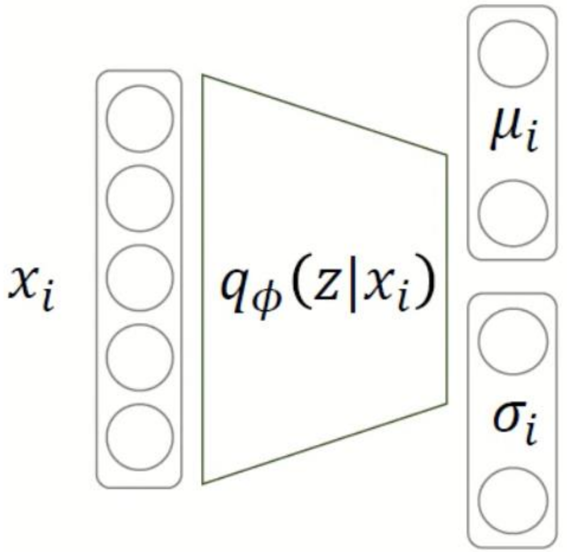
안정적으로 데이터 생성이 가능하다

Model description

Regularization part(KL):

p(z)는 우리가 가정하는 true distribution(간단한 형태)

Q(z|x)는 학습을 통해 얻어지는 gausssian distribution



Assumption 1

[Encoder : approximation class]

multivariate gaussian distribution with a diagonal covariance

$$q_\phi(z|x_i) \sim N(\mu_i, \sigma_i^2 I)$$

Assumption 2

[prior] multivariate normal distribution

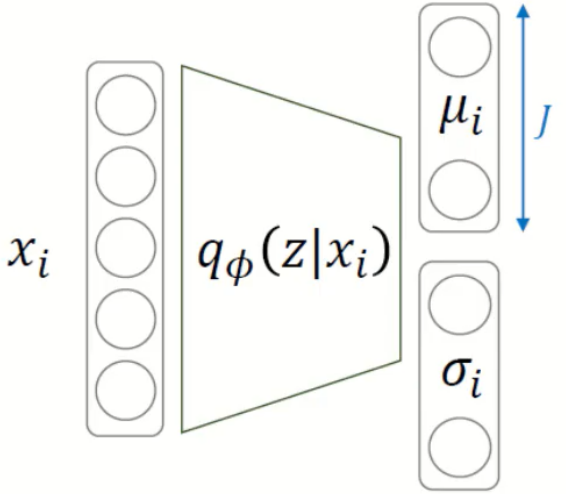
$$p(z) \sim N(0, I)$$

Model description

Regularization part(KL):

KLD

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



$$KL(q_\phi(z|x_i)||p(z)) = \frac{1}{2}\left\{tr\left(\sigma_i^2 I\right) + \mu_i^T\mu_i - J + ln\frac{1}{\prod_{j=1}^{J}\sigma_{i,j}^2}\right\}$$

$$= \frac{1}{2}\left\{\sum_{j=1}^{J}\sigma_{i,j}^2 + \sum_{j=1}^{J}\mu_{i,j}^2 - J - \sum_{j=1}^{J} ln(\sigma_{i,j}^2)\right\}$$

$N(0,I)$

$$= \frac{1}{2}\sum_{j=1}^{J}(\mu_{i,j}^2 + \sigma_{i,j}^2 - ln(\sigma_{i,j}^2) - 1)$$

KLD for multivariate normal distributions

$$D_{KL}(\mathcal{N}_0\|\mathcal{N}_1) = \frac{1}{2}\left(\mathrm{tr}\left(\Sigma_1^{-1}\Sigma_0\right) + (\mu_1-\mu_0)^{\mathsf{T}}\Sigma_1^{-1}(\mu_1-\mu_0) - k + \ln\left(\frac{\det\Sigma_1}{\det\Sigma_0}\right)\right)$$

Model description

Reconstruction Error:

C.1 Bernoulli MLP as decoder

In this case let $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ be a multivariate Bernoulli whose probabilities are computed from $\mathbf{z}$ with a fully-connected neural network with a single hidden layer:

$$\begin{aligned} \log p(\mathbf{x}|\mathbf{z}) &= \sum_{i=1}^{D} x_i \log y_i + (1 - x_i) \cdot \log(1 - y_i) \\ \text{where } \mathbf{y} &= f_\sigma(\mathbf{W}_2 \tanh(\mathbf{W}_1 \mathbf{z} + \mathbf{b}_1) + \mathbf{b}_2) \end{aligned} \tag{11}$$

where $f_\sigma(.)$ is the elementwise sigmoid activation function, and where $\boldsymbol{\theta} = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2\}$ are the weights and biases of the MLP.

C.2 Gaussian MLP as encoder or decoder

In this case let encoder or decoder be a multivariate Gaussian with a diagonal covariance structure:

$$\begin{aligned} \log p(\mathbf{x}|\mathbf{z}) &= \log \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2 \mathbf{I}) \\ \text{where } \boldsymbol{\mu} &= \mathbf{W}_4 \mathbf{h} + \mathbf{b}_4 \\ \log \boldsymbol{\sigma}^2 &= \mathbf{W}_5 \mathbf{h} + \mathbf{b}_5 \\ \mathbf{h} &= \tanh(\mathbf{W}_3 \mathbf{z} + \mathbf{b}_3) \end{aligned} \tag{12}$$

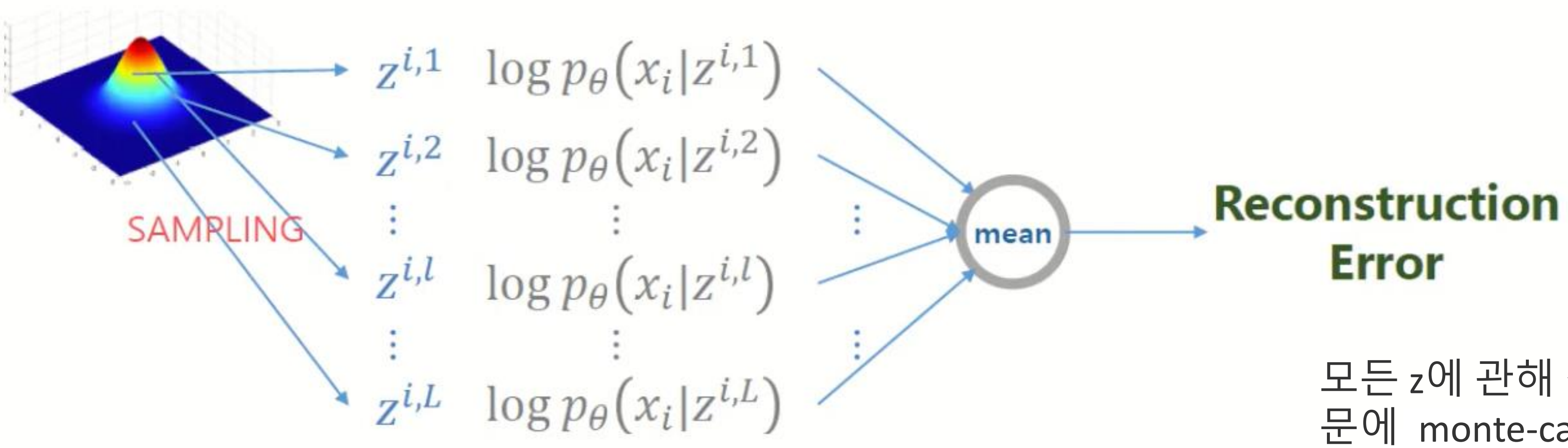
where $\{\mathbf{W}_3, \mathbf{W}_4, \mathbf{W}_5, \mathbf{b}_3, \mathbf{b}_4, \mathbf{b}_5\}$ are the weights and biases of the MLP and part of $\boldsymbol{\theta}$ when used as decoder. Note that when this network is used as an encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, then $\mathbf{z}$ and $\mathbf{x}$ are swapped, and the weights and biases are variational parameters $\boldsymbol{\phi}$.

Model description

Reconstruction Error:

$$\mathbb{E}_{q_\phi(\mathrm{z}|\mathrm{x_i})}[\log(p_\theta(\mathrm{x_i}|\mathrm{z}))] = \int \log(p_\theta(\mathrm{x_i}|\mathrm{z}))q_\phi(z|x_i)dz$$

Monte-carlo technique $\approx \frac{1}{L}\sum_{z^{i,l}} \log\left(p_\theta(\mathrm{x_i}|z^{i,l})\right)$

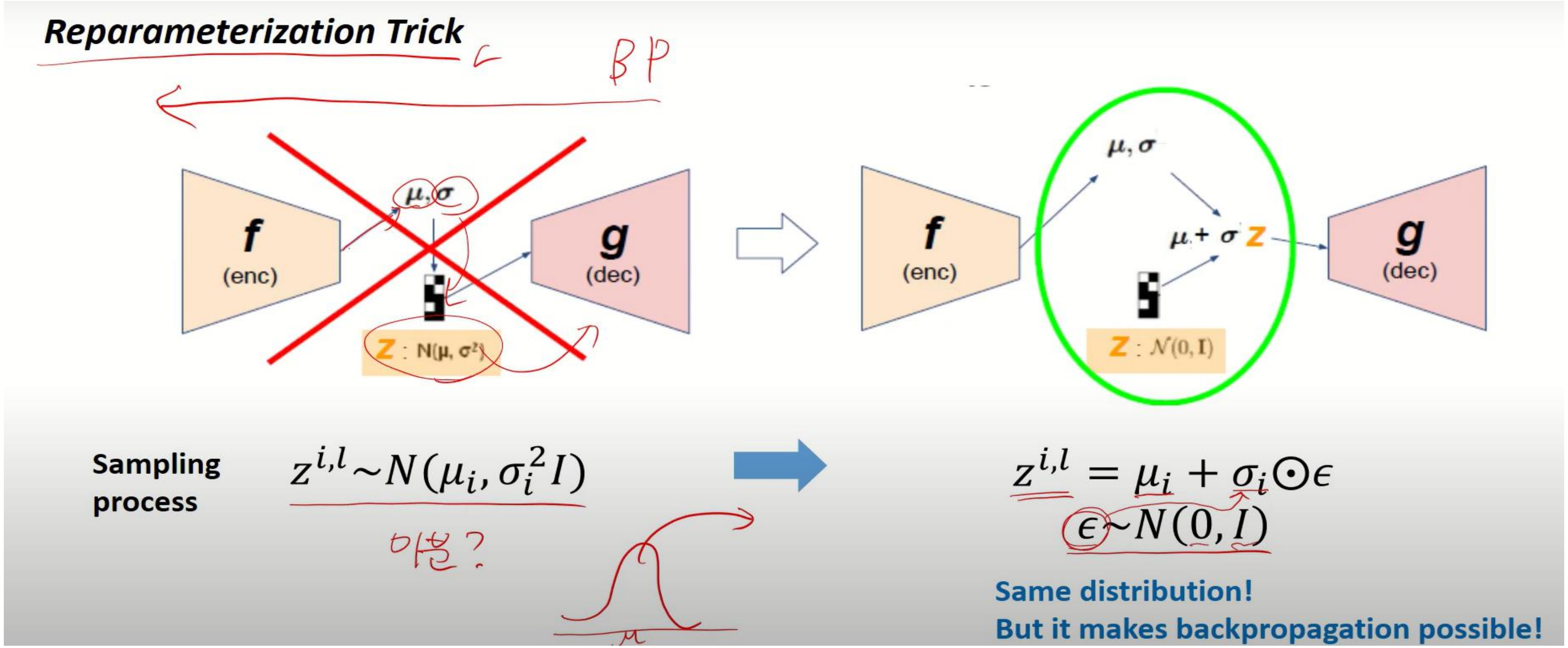


- L is the number of samples for latent vector
- Usually L is set to 1 for convenience

모든 z에 관해 적분을 하는 것은 쉽지 않기 때문에 monte-carloo technique 방법 또한 z를 매우 많이 샘플링해야 하기에 cost가 많이 든다. 따라서 L을 1로 하여 이를 단순화한다(batch 200이면 잘 작동한다고 논문에 작성됨)

Model description

Reparameterization Trick:



Z가 sampling하는 부분을 미분가능하게끔 변경하는 트릭, 미리 sampling을 해서 backpropagation을 사용하게끔 한다

Model description

Reconstruction Error(이미지 처리에서는 보통 bernoulli로 사용가능)

Assumptions

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

Reconstruction Error

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

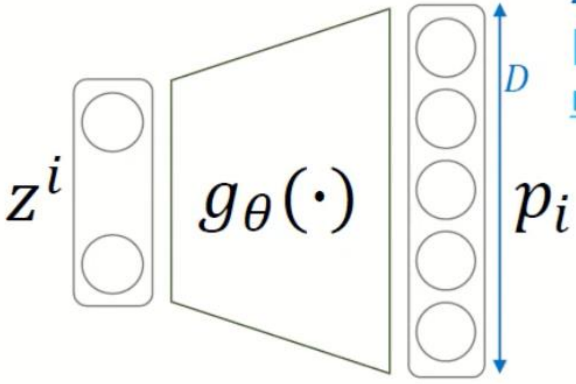
Monte-carlo technique

L=1

Assumption 3-1

[Decoder, likelihood]

multivariate bernoulli or gaussain distribution



$p_\theta(x_i|z^i) \sim Bernoulli(p_i)$

$$\log\left(p_\theta\left(x_i|z^i\right)\right) = \log\prod_{j=1}^{D} p_\theta\left(x_{i,j}|z^i\right) = \sum_{j=1}^{D} \log p_\theta\left(x_{i,j}|z^i\right)$$

$$= \sum_{j=1}^{D} \log p_{i,j}^{x_{i,j}}\left(1-p_{i,j}\right)^{1-x_{i,j}}$$

$p_{i,j}$: network output

$$= \sum_{j=1}^{D} x_{i,j}\log p_{i,j} + \left(1-x_{i,j}\right)\log\left(1-p_{i,j}\right)$$

Cross entropy

Model description

Reconstruction Error(가우시안인 case)

Assumption 3-2

[Decoder, likelihood]

multivariate bernoulli_or gaussain distribution

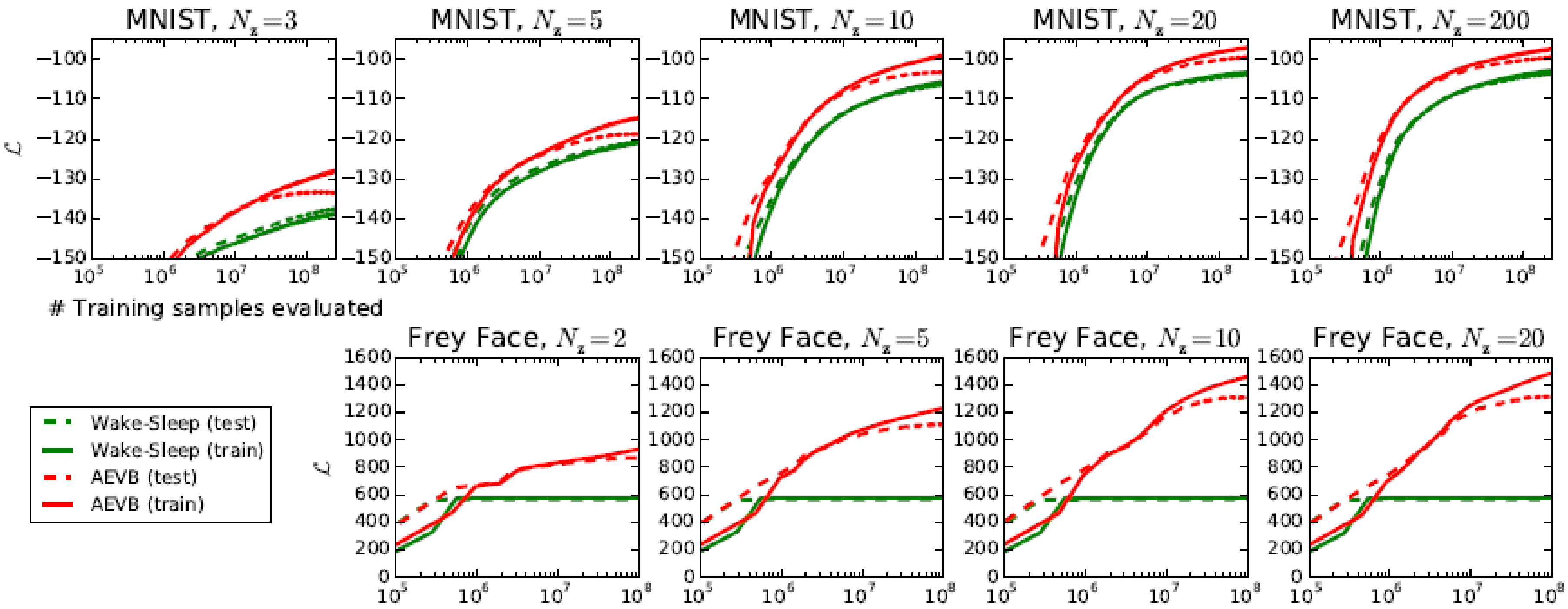
$$\log\left(p_\theta(x_i|z^i)\right) = \log(N(x_i;\mu_i,\sigma_i^2 I))$$

$$= -\sum_{j=1}^{D} \frac{1}{2}\log(\sigma_{i,j}^2) + \frac{(x_{i,j}-\mu_{i,j})^2}{2\sigma_{i,j}^2}$$

For gaussain distribution with identity covariance

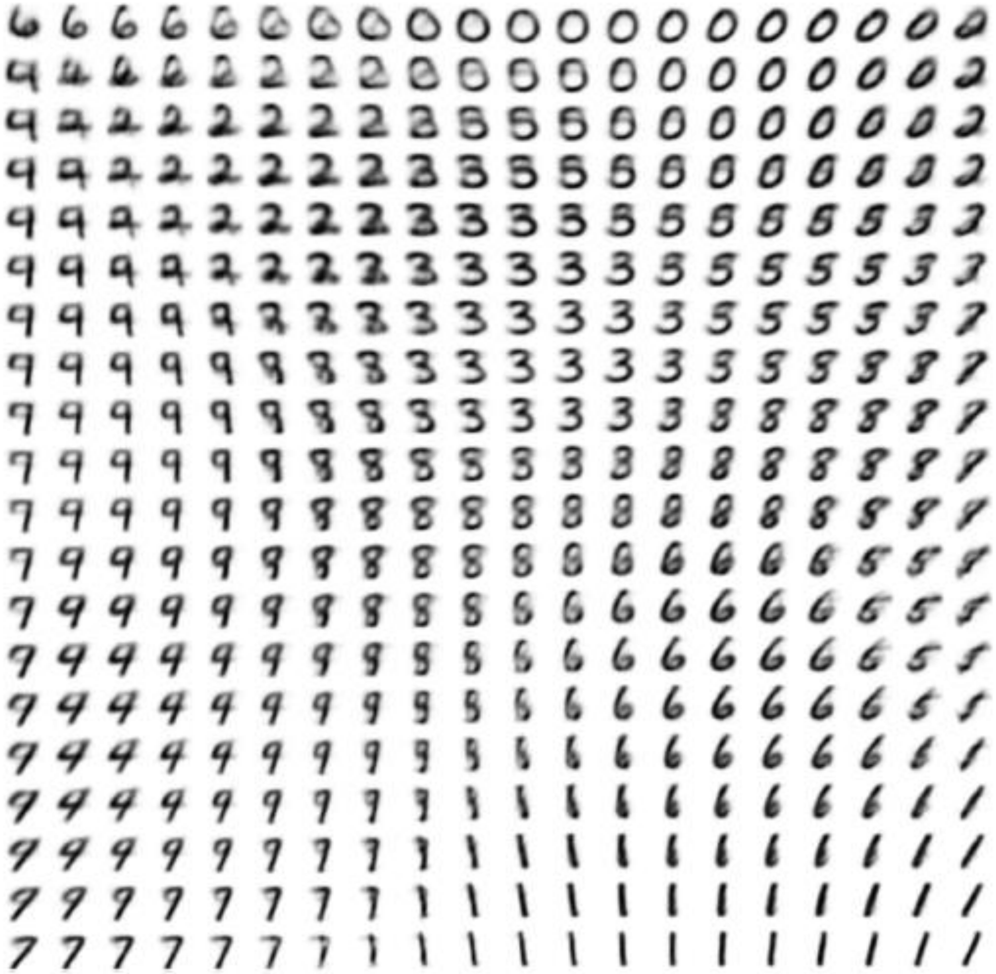
$$\log\left(p_\theta(x_i|z^i)\right) \propto -\sum_{j=1}^{D}(x_{i,j}-\mu_{i,j})^2$$ **Squared Error**

Experiment


MNIST, $N_z=3$
MNIST, $N_z=5$
MNIST, $N_z=10$
MNIST, $N_z=20$
MNIST, $N_z=200$
$\mathcal{L}$
Training samples evaluated
Frey Face, $N_z=2$
Frey Face, $N_z=5$
Frey Face, $N_z=10$
Frey Face, $N_z=20$
Wake-Sleep (test)
Wake-Sleep (train)
AEVB (test)
AEVB (train)

Experiment

(a) Learned Frey Face manifold



(b) Learned MNIST manifold

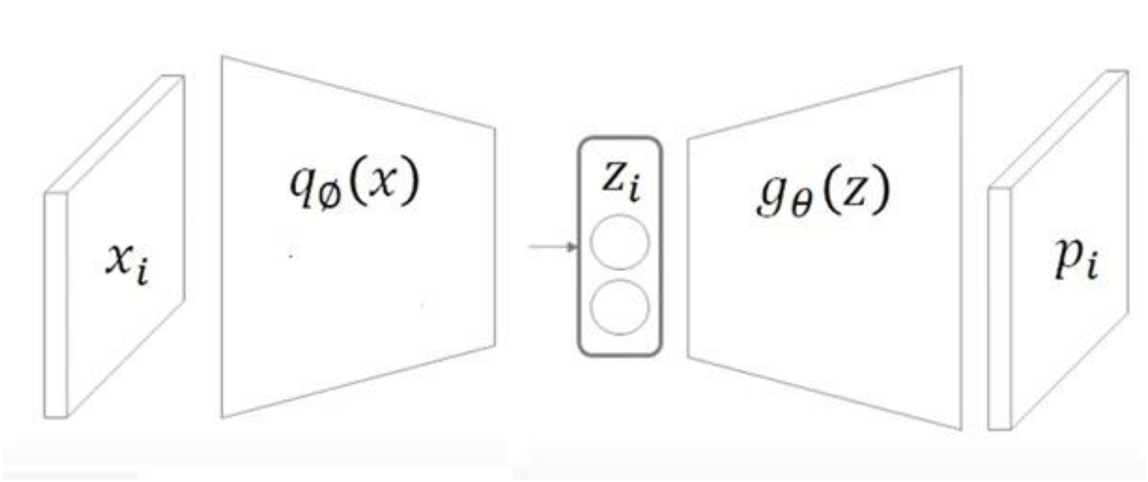
Experiment

(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Figure 5: Random samples from learned generative models of MNIST for different dimensionalities of latent space.

VGAE

Graph도 처리하고 싶다 -> encode와 decoder를 이에 맞춰 변경하자



Encoder: GCN(graph convolution network)

Decoder: simple inner product decoder

-> link prediction task in citation network

VGAE

Graph도 처리하고 싶다 -> encode와 decoder를 이에 맞춰 변경하자

(그래프 구조의 데이터는 불규칙하기 때문에 VAE의 개념을 그대로 적용할 수는 없다)

graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $N = |\mathcal{V}|$ nodes

adjacency matrix $\mathbf{A}$ of $\mathcal{G}$

degree matrix $\mathbf{D}$

stochastic latent variables $\mathbf{z}_i$,

$N \times D$ matrix $\mathbf{X}$

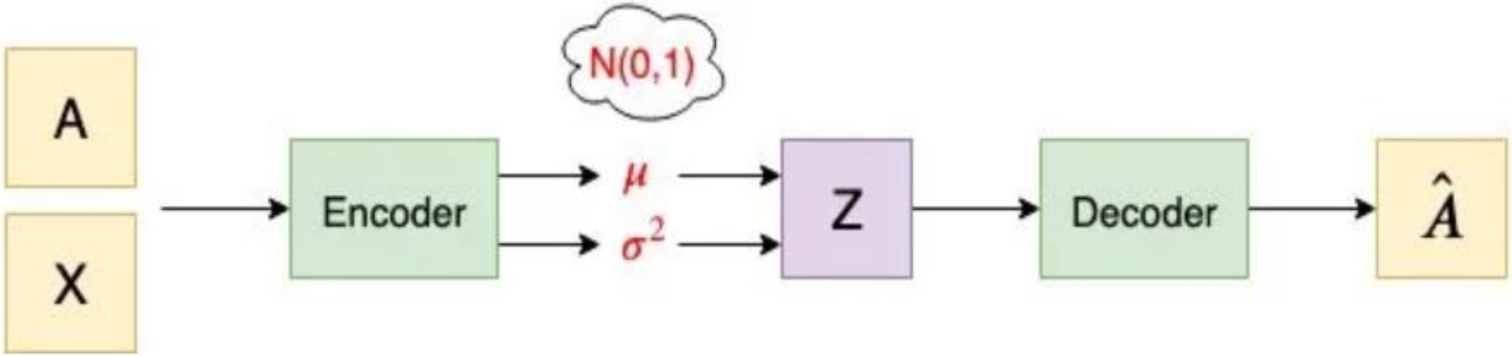


그림 10: Variational Graph Autoencoder의 아키텍처

VGAE

Inference model by two-layer GCN

two-layer GCN is defined as $\mathrm{GCN}(\mathbf{X}, \mathbf{A}) = \tilde{\mathbf{A}}\,\mathrm{ReLU}(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_0)\mathbf{W}_1$,

$\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix.

First Layer:

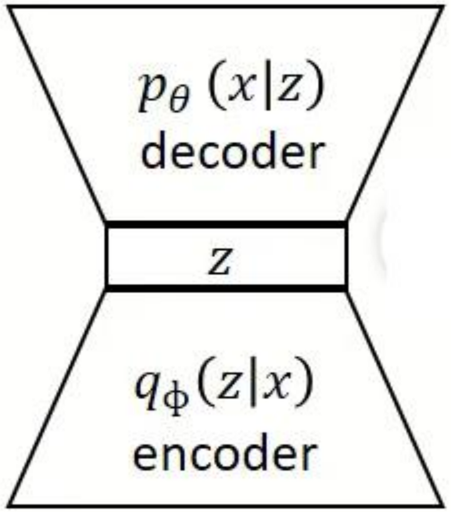
$$\bar{X} = GCN(X, A) = ReLU(\tilde{A}XW_0)$$

$$\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$$

Second layer: generate u and logσ²

$$\mu = GCN_\mu(X, A) = \tilde{A}\bar{X}W_1$$

$$log\sigma^2 = GCN_\sigma(X, A) = \tilde{A}\bar{X}W_1$$



VGAE

Reparameterization trick

two-layer GCN is defined as $\mathrm{GCN}(\mathbf{X}, \mathbf{A}) = \tilde{\mathbf{A}}\,\mathrm{ReLU}(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_0)\mathbf{W}_1$,

$$\mu = GCN_{\mu}(X, A) = \tilde{A}\bar{X}W_1$$

$$log\sigma^2 = GCN_{\sigma}(X, A) = \tilde{A}\bar{X}W_1$$

$$\Rightarrow$$

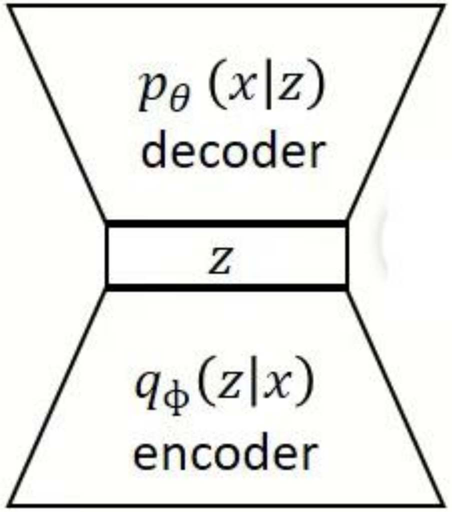
$$Z = \mu + \sigma * \epsilon$$

VGAE

Generative model(inner product decoder)

$$p(\mathbf{A} \mid \mathbf{Z}) = \prod_{i=1}^{N} \prod_{j=1}^{N} p(A_{ij} \mid \mathbf{z}_i, \mathbf{z}_j)\ , \quad \text{with} \quad p(A_{ij} = 1 \mid \mathbf{z}_i, \mathbf{z}_j) = \sigma(\mathbf{z}_i^\top \mathbf{z}_j)\ ,$$

After we get the latent variable *Z*, we want to find a way to learn the similarity of each row in the latent variable (because one row represents one vertex) to generate the output adjacency matrix.

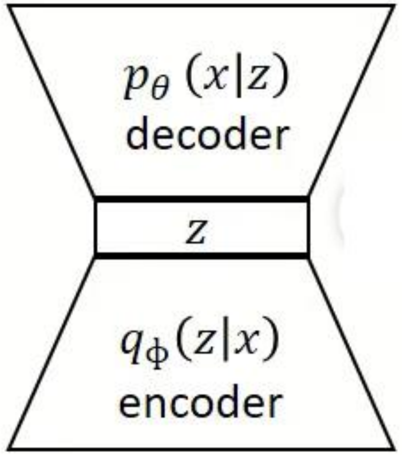


VGAE

Loss function

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}\left[\log p\left(\mathbf{A} \,|\, \mathbf{Z}\right)\right] - \mathrm{KL}\left[q(\mathbf{Z} \,|\, \mathbf{X}, \mathbf{A}) \,||\, p(\mathbf{Z})\right],$$

Reconstruction Error Regularization error



For very sparse A, it can be beneficial to
re-weight terms with $A_{ij} = 1$ in L or alternatively sub-sample terms with $A_{ij} = 0$. We choose the former for the following experiments.

We perform full-batch gradient decent

Experiment

Table 1: Link prediction task in citation networks. See [1] for dataset details.

Method	Cora		Citeseer		Pubmed	
	AUC	AP	AUC	AP	AUC	AP
SC [5]	84.6 ± 0.01	88.5 ± 0.00	80.5 ± 0.01	85.0 ± 0.01	84.2 ± 0.02	87.8 ± 0.01
DW [6]	83.1 ± 0.01	85.0 ± 0.00	80.5 ± 0.02	83.6 ± 0.01	84.4 ± 0.00	84.1 ± 0.00
GAE*	84.3 ± 0.02	88.1 ± 0.01	78.7 ± 0.02	84.1 ± 0.02	82.2 ± 0.01	87.4 ± 0.00
VGAE*	84.0 ± 0.02	87.7 ± 0.01	78.9 ± 0.03	84.1 ± 0.02	82.7 ± 0.01	87.5 ± 0.01
GAE	91.0 ± 0.02	92.0 ± 0.03	89.5 ± 0.04	89.9 ± 0.05	$\mathbf{96.4} \pm 0.00$	$\mathbf{96.5} \pm 0.00$
VGAE	$\mathbf{91.4} \pm 0.01$	$\mathbf{92.6} \pm 0.01$	$\mathbf{90.8} \pm 0.02$	$\mathbf{92.0} \pm 0.02$	94.4 ± 0.02	94.7 ± 0.02

Both VGAE and GAE achieve competitive results on the featureless task. Adding input features significantly improves predictive performance across datasets. A Gaussian prior is potentially a poor choice in combination with an inner product decoder, as the latter tries to push embeddings away from the zero-center (see Figure 1). Nevertheless, the VGAE model achieves higher predictive performance on both the Cora and the Citeseer dataset.