DATA
SCIENCE LAB

Auto-Encoder & Variational Auto-Encoder

23.03.09 / 8기 김남훈

CONTENTS

DATA
SCIENCE LAB

01. PRELIMINARY(사전지식)

1. PRELIMINARY(사전지식)

이전까지 해온 N.N.; notation 확인

1. Collect training data

$x = \{x_1, x_2, \dots, x_N\}$

$y = \{y_1, y_2, \dots, y_N\}$

$\mathcal{D} = \{(x_1, y_1), (x_2, y_2) \dots (x_N, y_N)\}$; N개의 데이터

2. Define functions

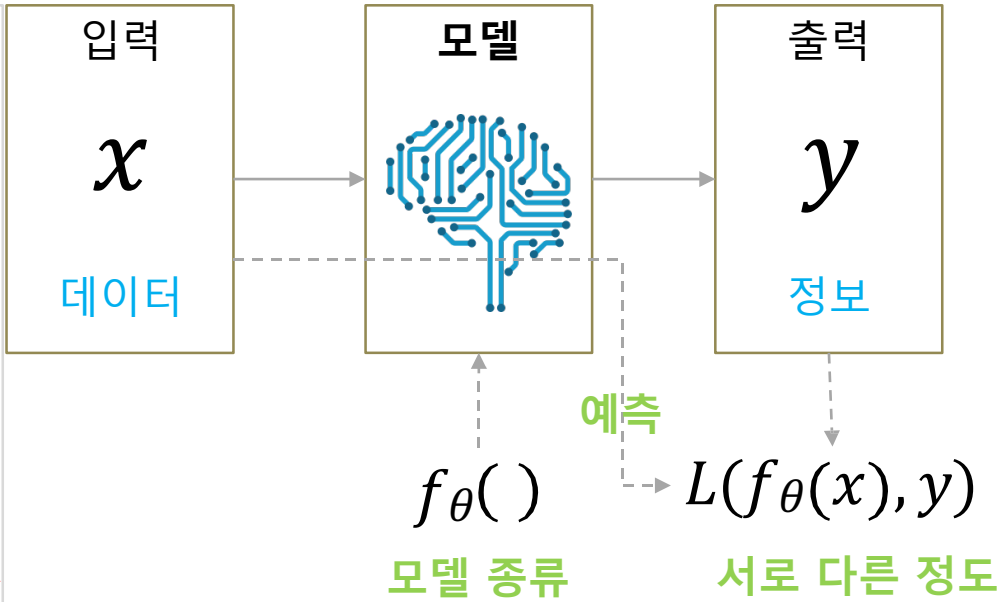
- Output : $f_\theta(x)$; $f()$: model, θ : weights & bias
- Loss : $L(f_\theta(x), y)$; $L()$: classical Loss-Function (MSE, Cross-Entropy)

3. Learning/Training

Find the optimal parameter

4. Predicting/Testing

Compute optimal function output



$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, L(f_\theta(x), y)$$

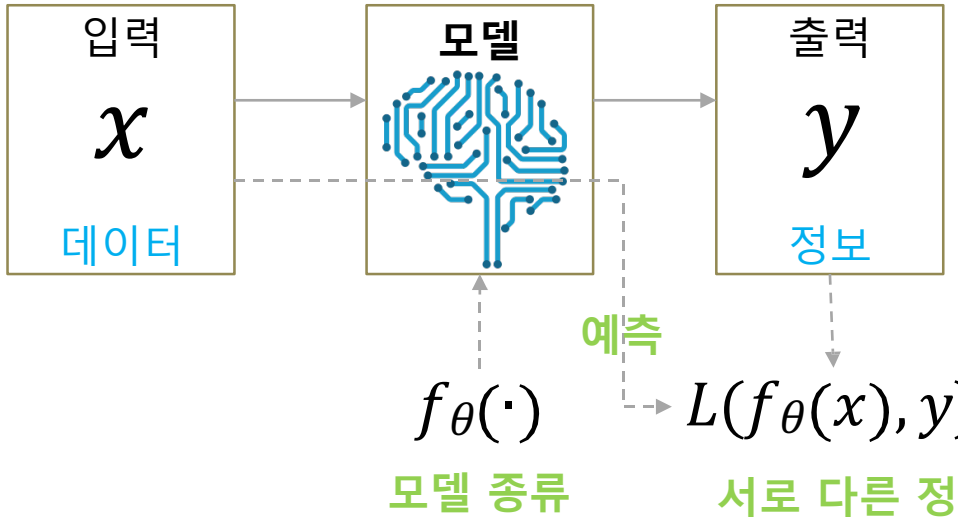
주어진 데이터를 제일 잘 설명하는 모델 찾기
= train-parameter 인 'θ(=weights & bias)' 찾기

$$y_{new} = f_{\theta^*}(x_{new})$$ 고정 입력, 고정 출력

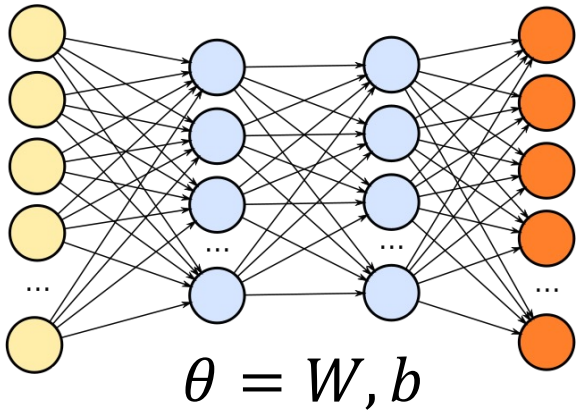
1. PRELIMINARY(사전지식)

이전까지 해온 N.N.; notation 확인

1. Collect training data
2. Define functions
3. Learning/Training
4. Predicting/Testing



$f_\theta(\,)$ Which Deep Neural Network Model ?



Train-parameter : weights & bias

$L(f_\theta(x), y)$ $L(f_\theta(x), y) = \sum_i L(f_\theta(x_i), y_i)$

Assumption 1.

Total loss of DNN over training samples is the **sum of loss for each training sample(1~N)**

Assumption 2.

Loss for each training example is a function of **final output** of DNN

Backpropagation을 통해 DNN학습을 학습 시키기 위한 조건들

1. PRELIMINARY(사전지식)

이전까지 해온 N.N.; notation 확인

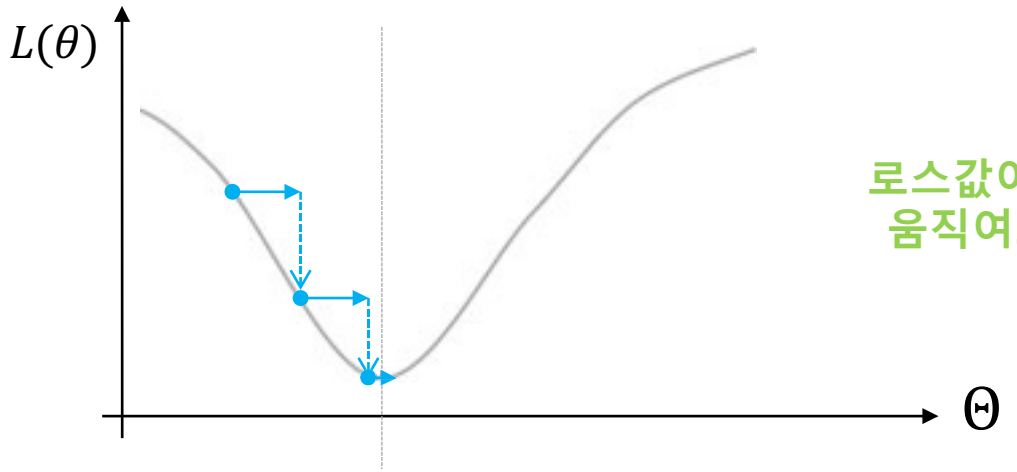
1. Collect training data
2. Define functions
3. Learning/Training
4. Predicting/Testing

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} L(f_\theta(x), y)$$

Gradient Descent

Iterative Method $\quad \theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} L(f_\theta(x), y) = \underset{\theta \in \Theta}{\operatorname{argmin}} L(\theta)$

Questions	*Strategies*
How to update $\theta \rightarrow \theta + \Delta\theta$	Only if $L(\theta + \Delta\theta) < L(\theta)$
When we stop to search??	If $L(\theta + \Delta\theta) == L(\theta)$



로스값이 줄어드는 방향으로 계속 이동하고,
움직여도 로스값이 변함 없을 경우 멈춘다

1. PRELIMINARY(사전지식)

이전까지 해온 N.N.; notation 확인

1. Collect training data
2. Define functions
3. Learning/Training
4. Predicting/Testing

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} L(f_\theta \ x \ , y)$$

Gradient Descent

Questions	*Strategies*
How to update $\theta \rightarrow \theta + \Delta\theta$	Only if $L(\theta + \Delta\theta) < L(\theta)$
When we stop to search??	If $L(\theta + \Delta\theta) == L(\theta)$
How to find $\Delta\theta$ so that $L(\theta + \Delta\theta) < L(\theta)$?	$\Delta\theta = -\eta\nabla L$, where $\eta > 0$

Taylor Expansion → $L(\theta + \Delta\theta) = L(\theta) + \nabla L \cdot \Delta\theta + second\ derivative + third\ derivative + \cdots$

Approximation → $L(\theta + \Delta\theta) \approx L(\theta) + \nabla L \cdot \Delta\theta$ 더 많은 차수를 사용할 수록 더 넓은 지역을 작은 오차로 표현 가능

$L(\theta + \Delta\theta) - L(\theta) = \Delta L = \nabla L \cdot \Delta\theta$

If $\Delta\theta = -\eta\Delta L$, then $\Delta L = -\eta\|\nabla L\|^2 < 0$, where $\eta > 0$ and called learning rate

∇L is gradient of L and indicates the steepest increasing direction of L

Learning rate를 사용하여 조금씩 파라미터 값을 바꾸는 것은 로스 함수의 1차 미분항까지만 사용했기에 아주 좁은 영역에서만 감소 방향이 정확하기 때문이다.

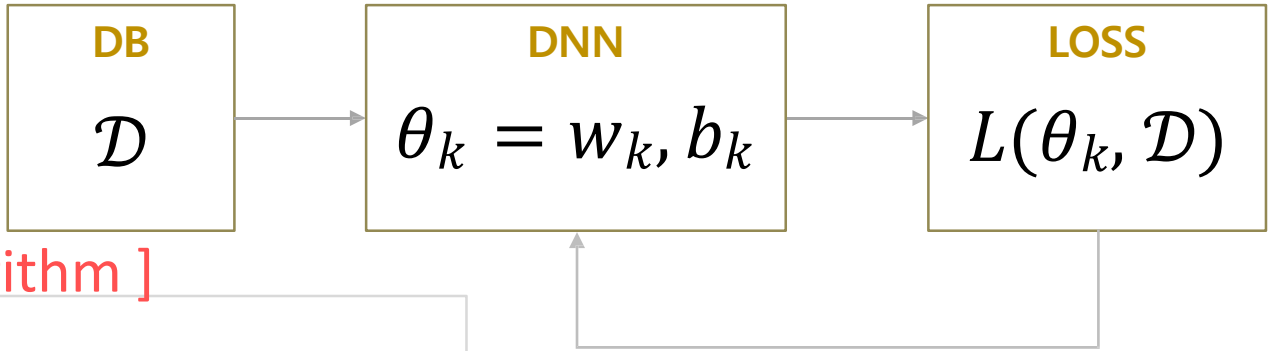
1. PRELIMINARY(사전지식)

이전까지 해온 N.N.; notation 확인

1. Collect training data
2. Define functions
3. Learning/Training
4. Predicting/Testing

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} L(f_\theta(x), y)$$

Gradient Descent + Backpropagation



$$\theta_{k+1} = \theta_k - \eta \nabla L(\theta_k, \mathcal{D})$$

$$w_{k+1}^l = w_k^l - \eta \nabla_{w_k^l} L(\theta_k, \mathcal{D})$$

$$b_{k+1}^l = b_k^l - \eta \nabla_{b_k^l} L(\theta_k, \mathcal{D})$$

특정 레이어에서 파라미터 갱신식

로스함수의 미분값이 딥뉴럴넷 학습에서 제일 중요!!

[Backpropagation Algorithm]

1. Error at the output layer

$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

- C : Cost (Loss)
- a : final output of DNN
- $\sigma(\cdot)$: activation function

2. Error relationship between two adjacent layers

$$\delta^l = \sigma'(z^l) \odot \left((w^{l+1})^T \delta^{l+1} \right)$$

3. Gradient of C in terms of bias

$$\nabla_{b^l} C = \delta^l$$

4. Gradient of C in terms of weight

$$\nabla_{W^l} C = \delta^l (a^{l-1})^T$$

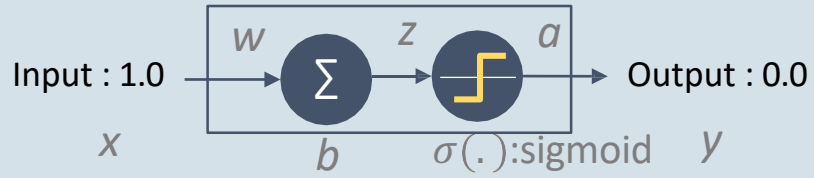
1. PRELIMINARY(사전지식)

Loss Function에 대한 접근(1)

Q. Classical Loss-Function으로 보았을 때, 언제 MSE(=Mean-Square-Error), CE(=Cross-Entropy)를 사용할까 ?

Type 1 : Mean Square Error / Quadratic loss

OBJECT



$$C = \|a - y\|^2/2 = a^2/2$$

$$\nabla_a C = (a - y)$$

$$\delta = \nabla_a C \odot \sigma'(z) = (a - y)\sigma'(z)$$

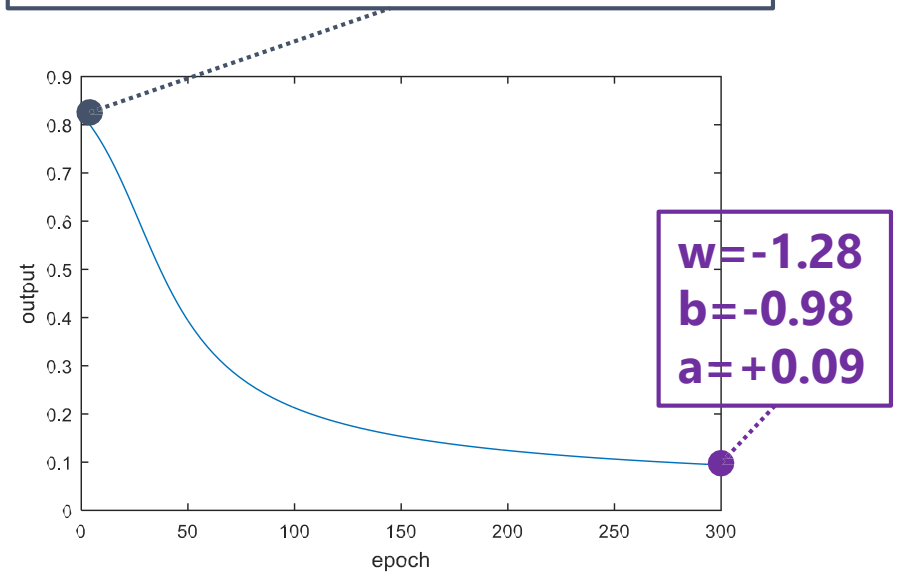
$$\frac{\partial C}{\partial w} = x\delta = \delta$$

$$w = w - \eta\delta$$

$$\frac{\partial C}{\partial b} = \delta$$

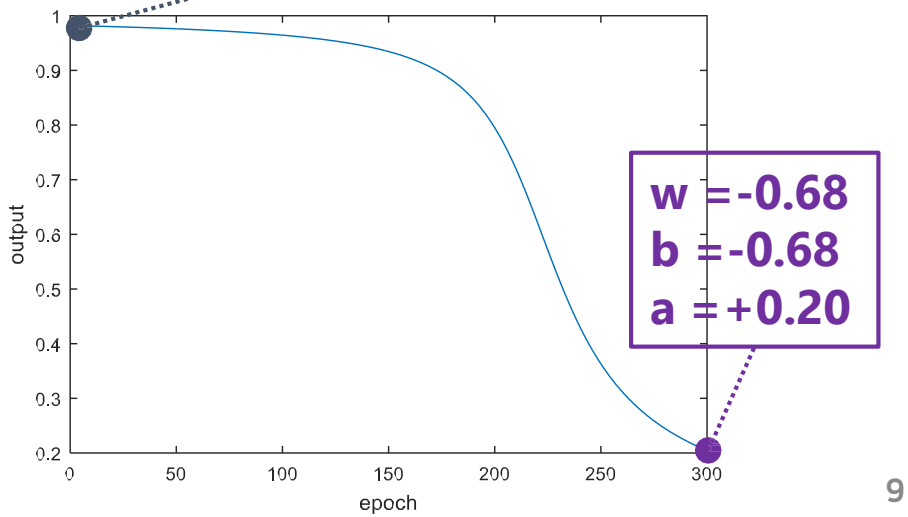
$$b = b - \eta\delta$$

$w_0 = +0.6$, $b_0 = +0.9$, $a_0 = +0.82$



w=-1.28
b=-0.98
a=+0.09

$w_0 = +2.0$, $b_0 = +2.0$, $a_0 = +0.98$



w =-0.68
b =-0.68
a =+0.20

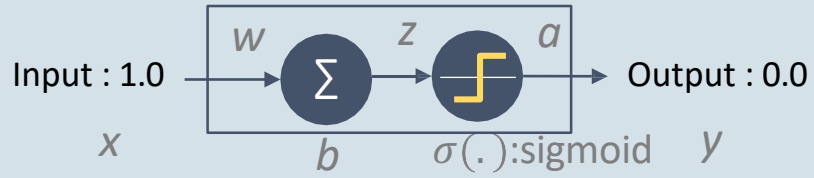
1. PRELIMINARY(사전지식)

Loss Function에 대한 접근(1)

Q. Classical Loss-Function으로 보았을 때, 언제 MSE(=Mean-Square-Error), CE(=Cross-Entropy)를 사용할까 ?

Type 1 : Mean Square Error / Quadratic loss

OBJECT



$$C = \|a - y\|^2/2 = a^2/2$$

$$\nabla_a C = (a - y)$$

$$\delta = \nabla_a C \odot \sigma'(z) = (a - y)\sigma'(z)$$

$$\frac{\partial C}{\partial w} = x\delta = \delta$$

$$w = w - \eta\delta$$

$$\frac{\partial C}{\partial b} = \delta$$

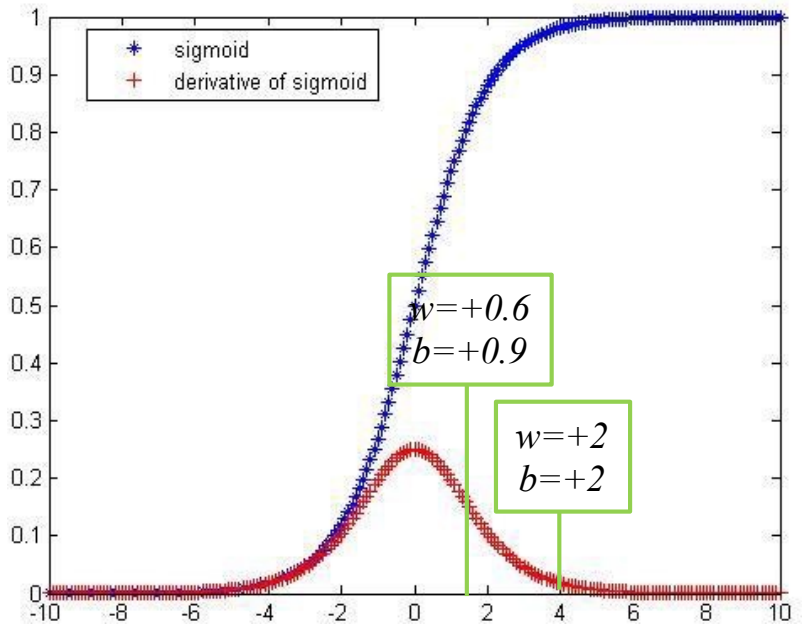
$$b = b - \eta\delta$$

Learning slow means are $\partial C / \partial w, \partial C / \partial b$ small !!

Why they are small???

$$\frac{\partial C}{\partial w} = x\delta = xa\sigma'(z) = a\sigma'(z)$$

$$\frac{\partial C}{\partial b} = \delta = a\sigma'(z)$$



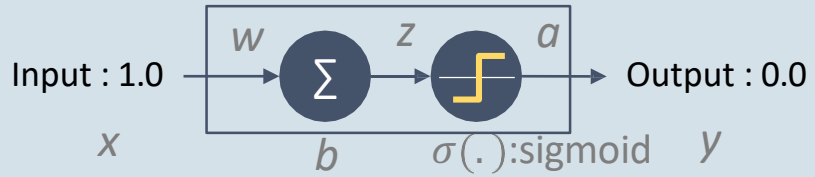
1. PRELIMINARY(사전지식)

Loss Function에 대한 접근(1)

Q. Classical Loss-Function으로 보았을 때, 언제 MSE(=Mean-Square-Error), CE(=Cross-Entropy)를 사용할까 ?

Type 2 : Cross Entropy

OBJECT



MSE와는 달리 CE는 출력 레이어에서의 에러값에 activation function의 미분값이 곱해지지 않아 gradient vanishing problem에서 좀 더 자유롭다.
(학습이 좀 더 빨리 된다)

그러나 레이어가 여러 개가 사용될 경우에는 결국 activation function의 미분값이 계속해서 곱해지므로 gradient vanishing problem에서 완전 자유로울 수 없다.

ReLU는 미분값이 1 혹은 0이므로 이러한 관점에서 훌륭한 activation function이다.

$$C = -[y\,ln\,a + (1-y)ln(1-a)\,]$$

$$\nabla_a C = -\frac{y}{a} - (1-y)\frac{-1}{1-a} = \frac{y-a}{(1-a)a}$$

$$= -\frac{y}{a} + \frac{(1-y)}{1-a}$$

$$= \frac{-(1-a)y}{(1-a)a} + \frac{(1-y)a}{(1-a)a}$$

$$= \frac{-y+ay+a-ay}{(1-a)a}$$

$$= \frac{a-y}{(1-a)a}$$

$$\sigma'(z) = \frac{\partial a}{\partial z} = \sigma'(z) = \big(1-\sigma(z)\big)\sigma(z) = (1-a)a$$

$$\delta_{MSE} = (a-y)\sigma'(z) \dashrightarrow \delta_{CE} = \nabla_a C \odot \sigma'(z^L) = \frac{a-y}{(1-a)a}(1-a)a = a-y$$

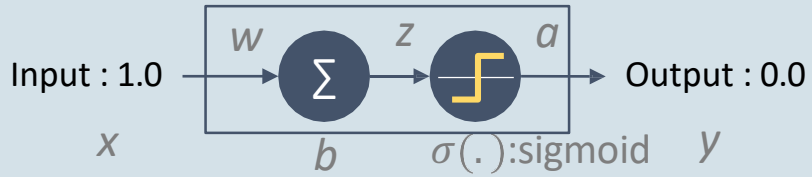
1. PRELIMINARY(사전지식)

Loss Function에 대한 접근(1)

Q. Classical Loss-Function으로 보았을 때, 언제 MSE(=Mean-Square-Error), CE(=Cross-Entropy)를 사용할까 ?

Type 2 : Cross Entropy

OBJECT



$$C = -[y\,ln\,a + (1-y)ln(1-a)\,]$$

$$\delta = a - y$$

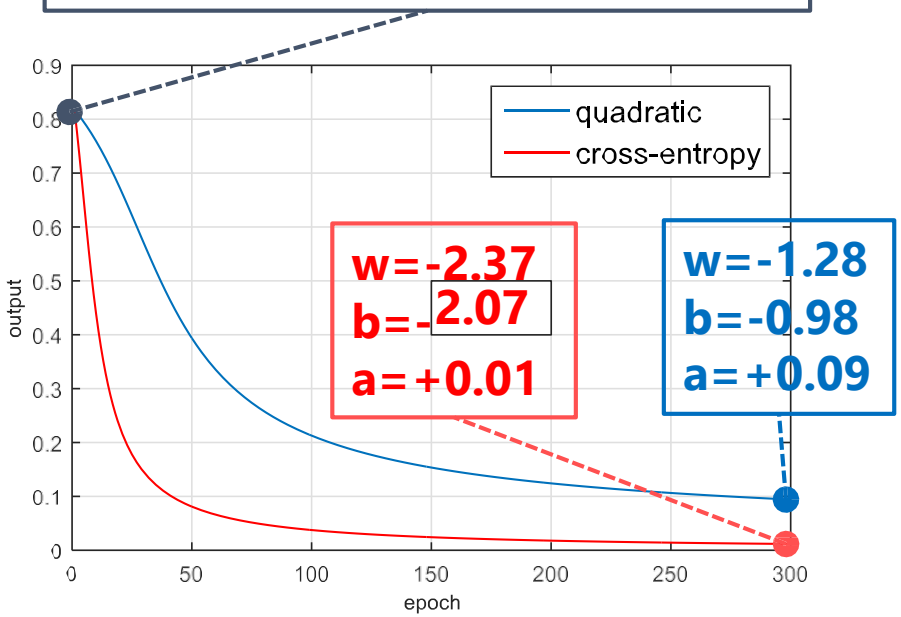
$$\frac{\partial C}{\partial w} = x\delta = \delta$$

$$w = w - \eta\delta$$

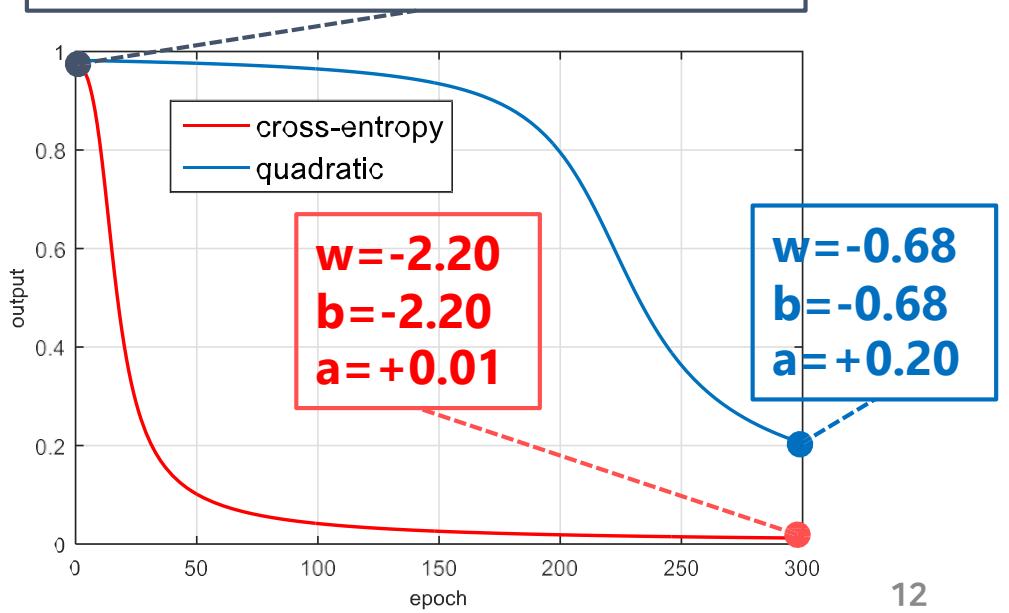
$$\frac{\partial C}{\partial b} = \delta$$

$$b = b - \eta\delta$$

w_0=+0.6, b_0=+0.9, a_0=+0.82



w_0=+2.0, b_0=+2.0, a_0=+0.98



1. PRELIMINARY(사전지식)

M.L.E.로 해석한 N.N.

1. Collect training data

$$x = \{x_1, x_2, \dots, x_N\}$$
$$y = \{y_1, y_2, \dots, y_N\}$$
$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2) \dots (x_N, y_N)\}$$

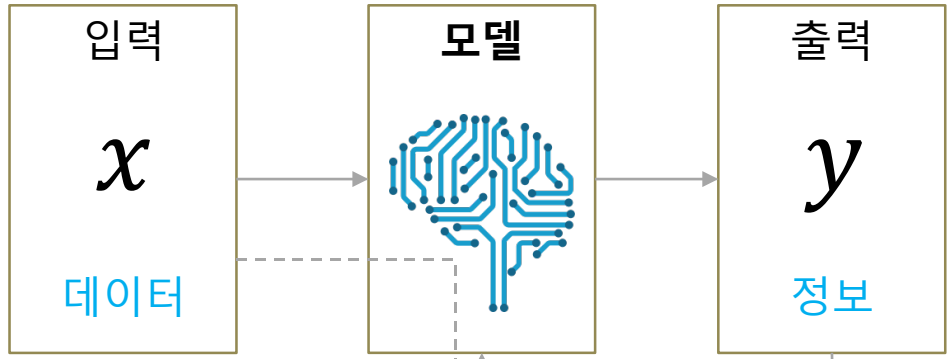
2. Define functions
   - Output : $f_\theta(x)$
   - Loss : $-\log(p(y|f_\theta(x)))$

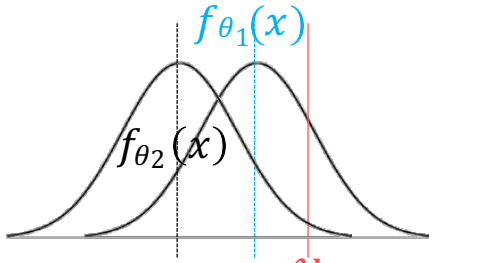
3. Learning/Training

   Find the optimal parameter

4. Predicting/Testing

   Compute optimal function output





$$p(y|f_{\theta_2}(x)) < p(y|f_{\theta_1}(x))$$

예측

$f_\theta(\cdot)$ → $p(y|f_\theta(x))$

모델 종류

정해진 확률분포에서 출력이 나올 확률

$$\theta^* = \underset{\theta}{\operatorname{argmin}}[-\log(p(y|f_\theta(x)))]$$

주어진 데이터를 제일 잘 설명하는 모델 찾기

$$y_{new} \sim p(y|f_{\theta^*}(x_{new}))$$

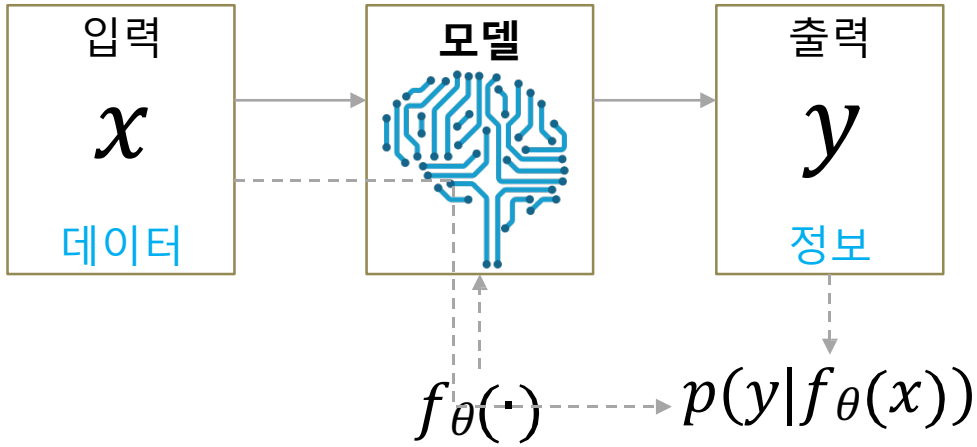
고정 입력, 고정/다른 출력

1. PRELIMINARY(사전지식)

M.L.E.로 해석한 N.N.

Back to Machine Learning Problem

1. Collect training data
2. Define functions
3. Learning/Training
4. Predicting/Testing



i.i.d Condition on $p(y|f_\theta(x))$

Assumption 1 : Independence

All of our data is independent of each other

$$p(y|f_\theta(x)) = \varsigma_i \; p_{D_i}(y|f_\theta(x_i))$$

Assumption 2: Identical Distribution

Our data is identically distributed

$$p(y|f_\theta(x)) = \varsigma_i \; p(y|f_\theta(x_i))$$

$$-\log\left(p(y|f_\theta(x))\right) = -\sum_i \log\left(p(y_i|f(x_i))\right)$$

Assumption 1. OK

Total loss of DNN over training samples is the sum of loss for each training sample

Assumption 2. OK

Loss for each training example is a function of final output of DNN

1. PRELIMINARY(사전지식)

Loss Function에 대한 접근(2)

Q. MLE으로 보았을 때, 언제 MSE(=Mean-Square-Error), CE(=Cross-Entropy)를 사용할까 ?

Univariate cases

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

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

1. PRELIMINARY(사전지식)

Loss Function에 대한 접근(2)

Q. MLE으로 보았을 때, 언제 MSE(=Mean-Square-Error), CE(=Cross-Entropy)를 사용할까 ?

Multivariate cases

Gaussian distribution

$$f_\theta(x_i) = \mu_i, \Sigma_i = I \qquad -\log\big(p(y_i|f_\theta(x_i))\big)$$

$$p(y_i|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{n/2}|\Sigma_i|^{1/2}} \exp\left(-\frac{(y_i-\mu_i)^T\Sigma_i^{-1}(y_i-\mu_i)}{2}\right)$$

$$\log(p(y_i|\mu_i, \Sigma_i)) = \log\frac{1}{(2\pi)^{n/2}|\Sigma_i|^{1/2}} - \frac{(y_i-\mu_i)^T\Sigma_i^{-1}(y_i-\mu_i)}{2}$$

$$-\log(p(y_i|\mu_i)) = -\log\frac{1}{(2\pi)^{n/2}} + \frac{\|y_i-\mu_i\|_2^2}{2}$$

$$-\log(p(y_i|\mu_i)) \propto \frac{\|y_i-\mu_i\|_2^2}{2} = \frac{\|y_i-f_\theta(x_i)\|_2^2}{2}$$

Mean Squared Error

Categorical distribution

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

$$p(y_i|p_i) = \varsigma_{j=1}^{n} p_{i,j}^{y_{i,j}} (1-p_{i,j})^{1-y_{i,j}}$$

$$\log(p(y_i|p_i)) = \sigma_{j=1}^{n}\big(y_{i,j}\log p_{i,j} + (1-y_{i,j})\log(1-p_{i,j})\big)$$

$$-\log(p(y_i|p_i)) = -\sigma_{j=1}^{n}\big(y_{i,j}\log p_{i,j} + (1-y_{i,j})\log(1-p_{i,j})\big)$$

Cross-entropy

Also called Generalized Bernoulli or Multinoulli distribution

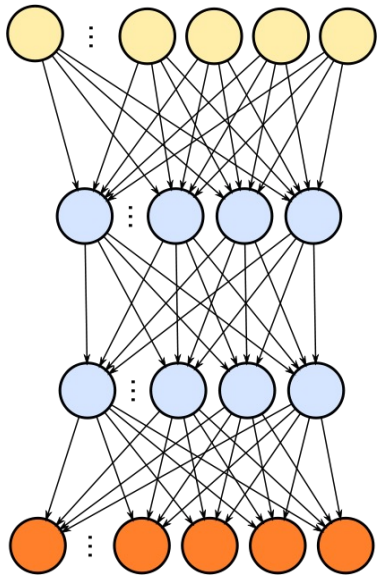
1. PRELIMINARY(사전지식)

Loss Function에 대한 접근(2)

Q. MLE으로 보았을 때, 언제 MSE(=Mean-Square-Error), CE(=Cross-Entropy)를 사용할까 ?

Distribution estimation

$p(y_i|x_i)$

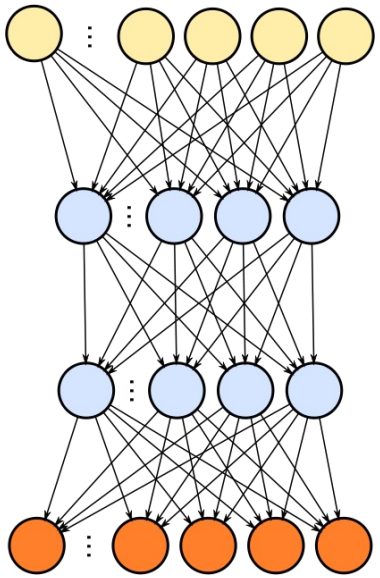


x_i

Likelihood값을 예측하는 것이 아니라,
Likelihood의 파라미터값을 예측하는 것이다.

Gaussian distribution

$f_\theta(x_i) = \mu_i$

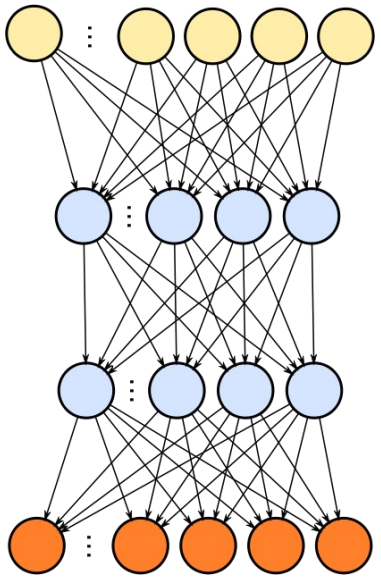


x_i

Mean Squared Error

Categorical distribution

$f_\theta(x_i) = p_i$



x_i

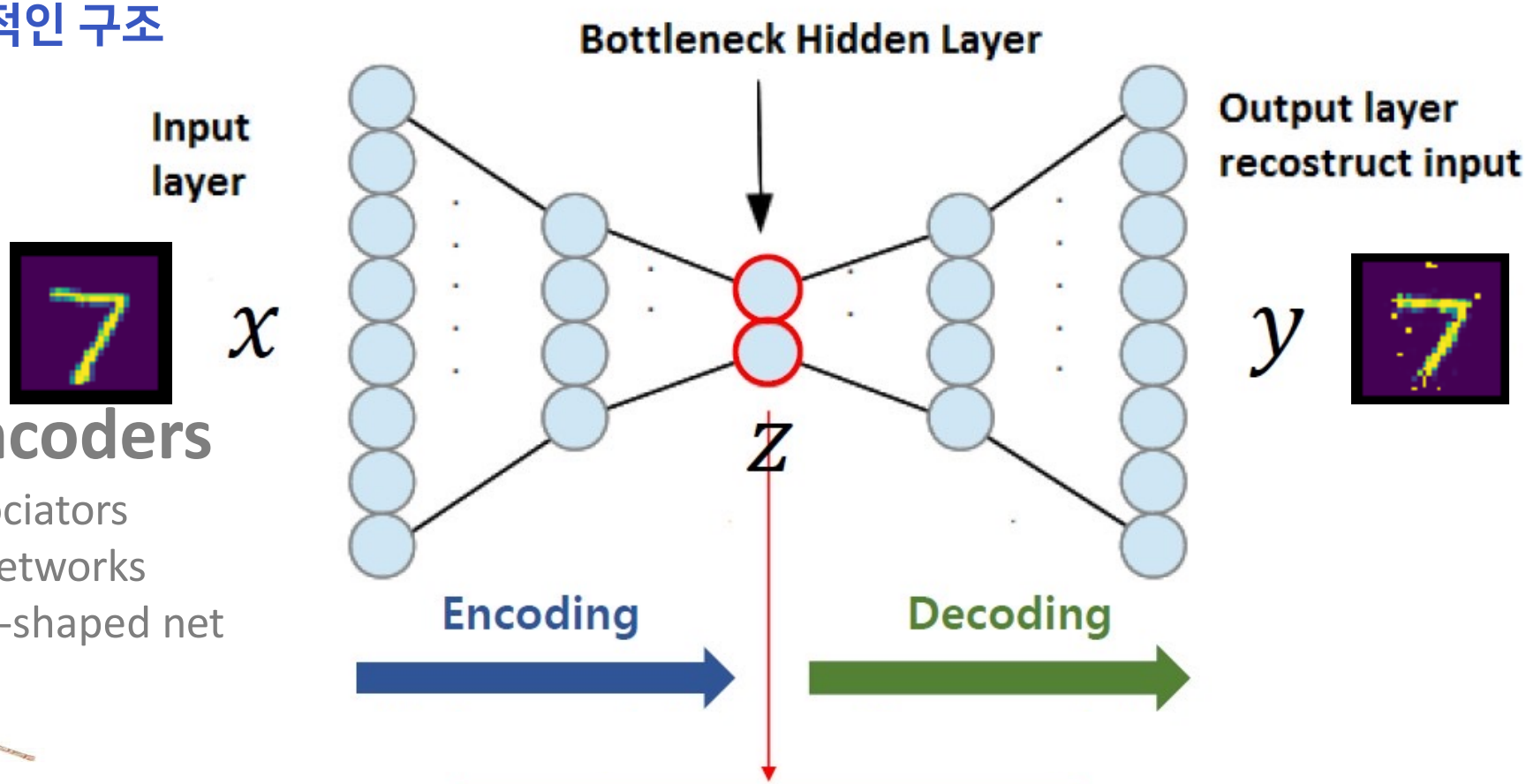
Cross-entropy

DATA
SCIENCE LAB

02. Auto-Encoder(=A.E.)

2. Auto-Encoder(=A.E.)

전체적인 구조



Autoencoders

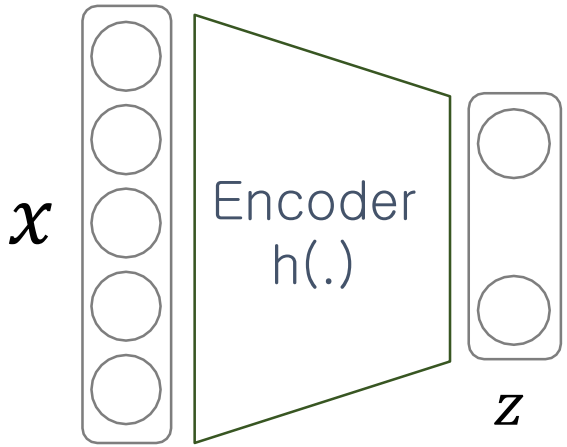
= Auto-associators
= Diabolo networks
= Sandglass-shaped net

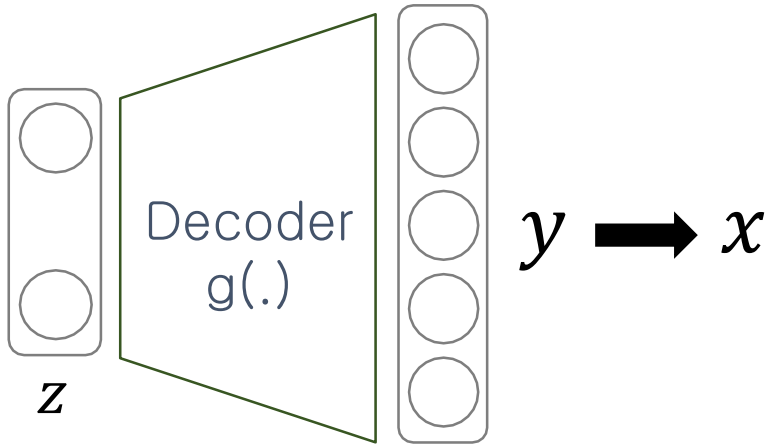
Diabolo

- Code
- Latent Variable
- Feature
- Hidden representation

2. Auto-Encoder(=A.E.)

사용목적 : Manifold-Learning; Encoder





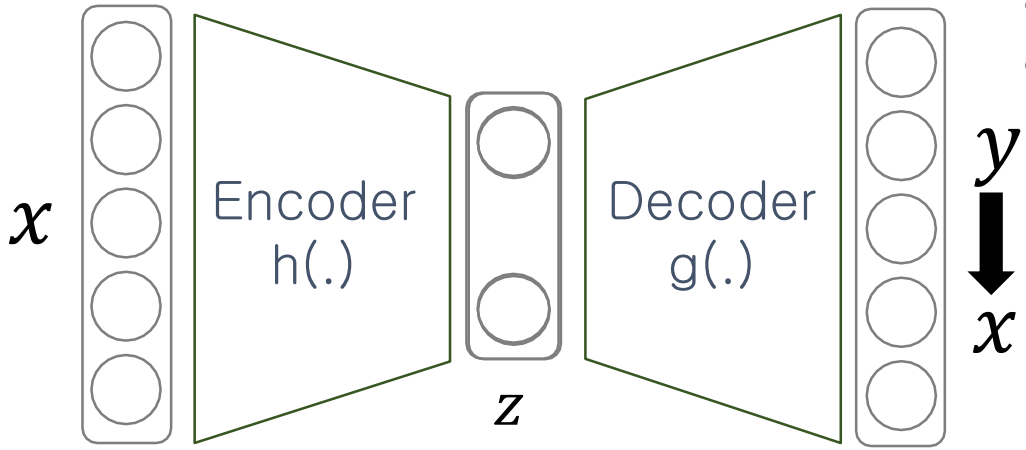
- 차원축소의 개념 : 임의의 벡터 혹은 행렬 형태의 데이터(X; input-data)를 어떤 벡터(Z; latent-vector)로 압축하고 싶다. 이 때, 입력데이터인 X를 어떤 벡터 Z로 압축시키는 N.N.을 Encoder라고 부른다.
- 잘 축소했는지의 기준 : 압축된 벡터(Z)를 통해 다시 원본 데이터(X)를 만들어 낼 수 있으면, 잘 압축된 것이다.
- 어떻게 구현할 것인가 ?

  Step 1. 뒷 단에 압축된 벡터(Z)인 latent-vector로 원본 데이터(X)를 예측하는 N.N.을 만든 뒤, 이를 Decoder라고 부른다.

  Step2. Decoder형태의 N.N.의 output값이 Encoder형태의 N.N.의 input값과 동일하도록 학습시킨다. 따라서 Unsupervised -> Supervised 문제로 바뀐다.

2. Auto-Encoder(=A.E.)

사용목적 : Manifold-Learning; Encoder



- 즉, Encoder인 N.N.을 잘 학습시키기 위해, Decoder인 N.N.이 도입된 상황 !
- Encoder와 Decoder를 붙여서 학습시킨 이후에는 다시 Encoder를 떼어서 Encoder부분에 학습된 가중치들을 이용하여, Manifold-Learning을 진행한다.

Manifold learning

- 비선형 차원 축소 방법을 의미
- 고차원 데이터가 있을 때 sample들을 잘 아우르는 subspace가 있을 것이라는 가정하에 학습을 진행

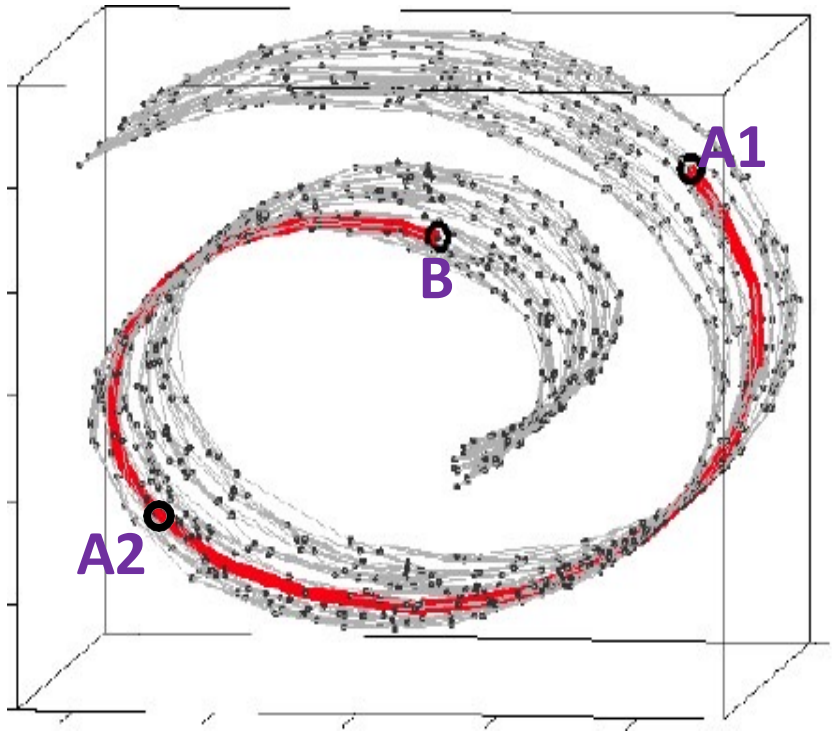
		기존 입력변수 어떻게 조합할 것인가?	
		선형 결합	비선형 결합
Y라벨을 고려하여 특징을 생성할 것인가?	Unsupervised	PCA MDS	KPCA Isomap LLE t-SNE AutoEncoder
	Supervised	LDA	KFDA

2. Auto-Encoder(=A.E.)

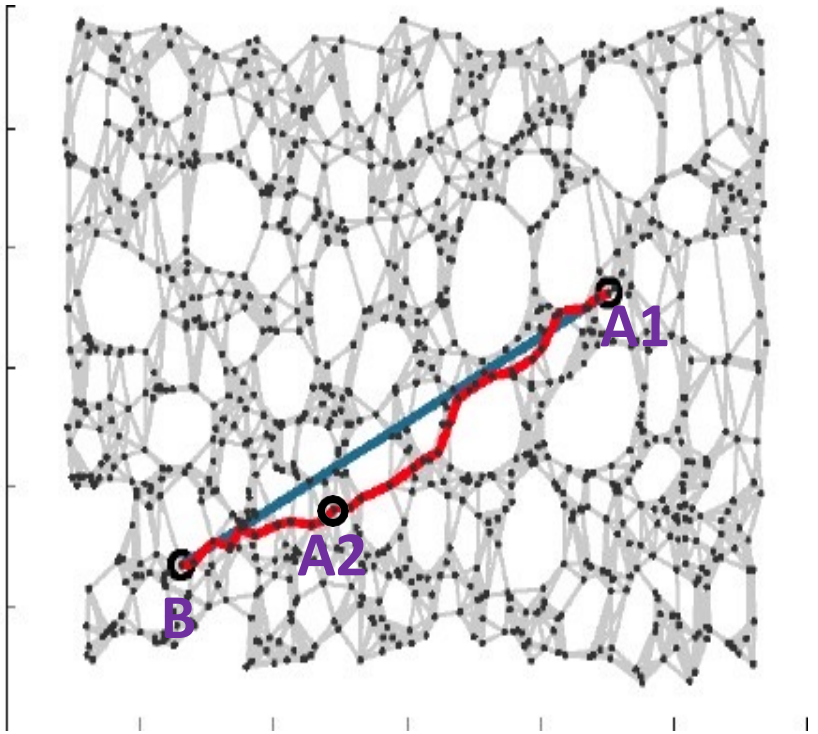
사용목적 : Manifold-Learning; Encoder

Remark. What is Manifold-Learning ?

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



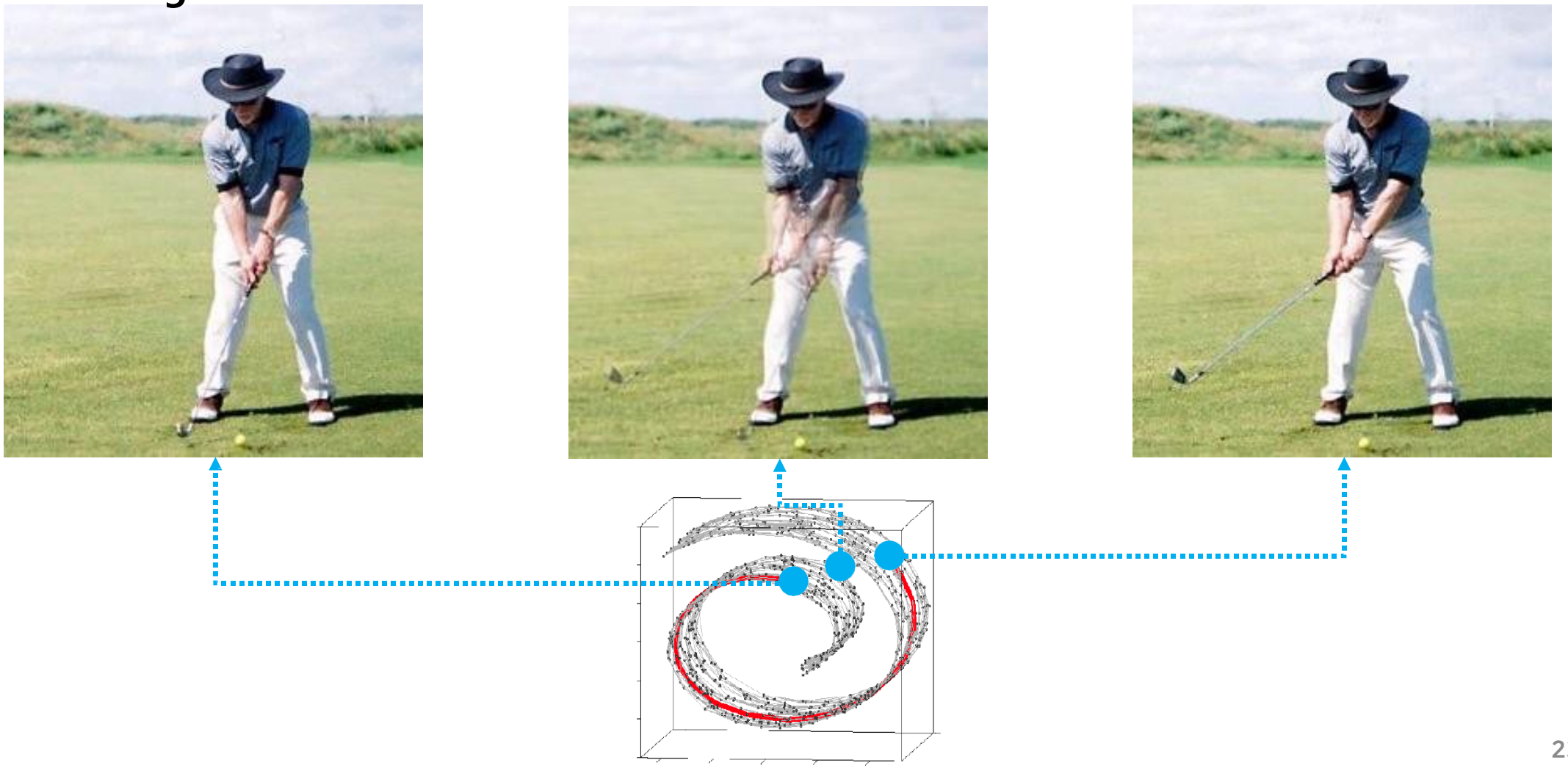
Distance in manifold

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

2. Auto-Encoder(=A.E.)

사용목적 : Manifold-Learning; Encoder

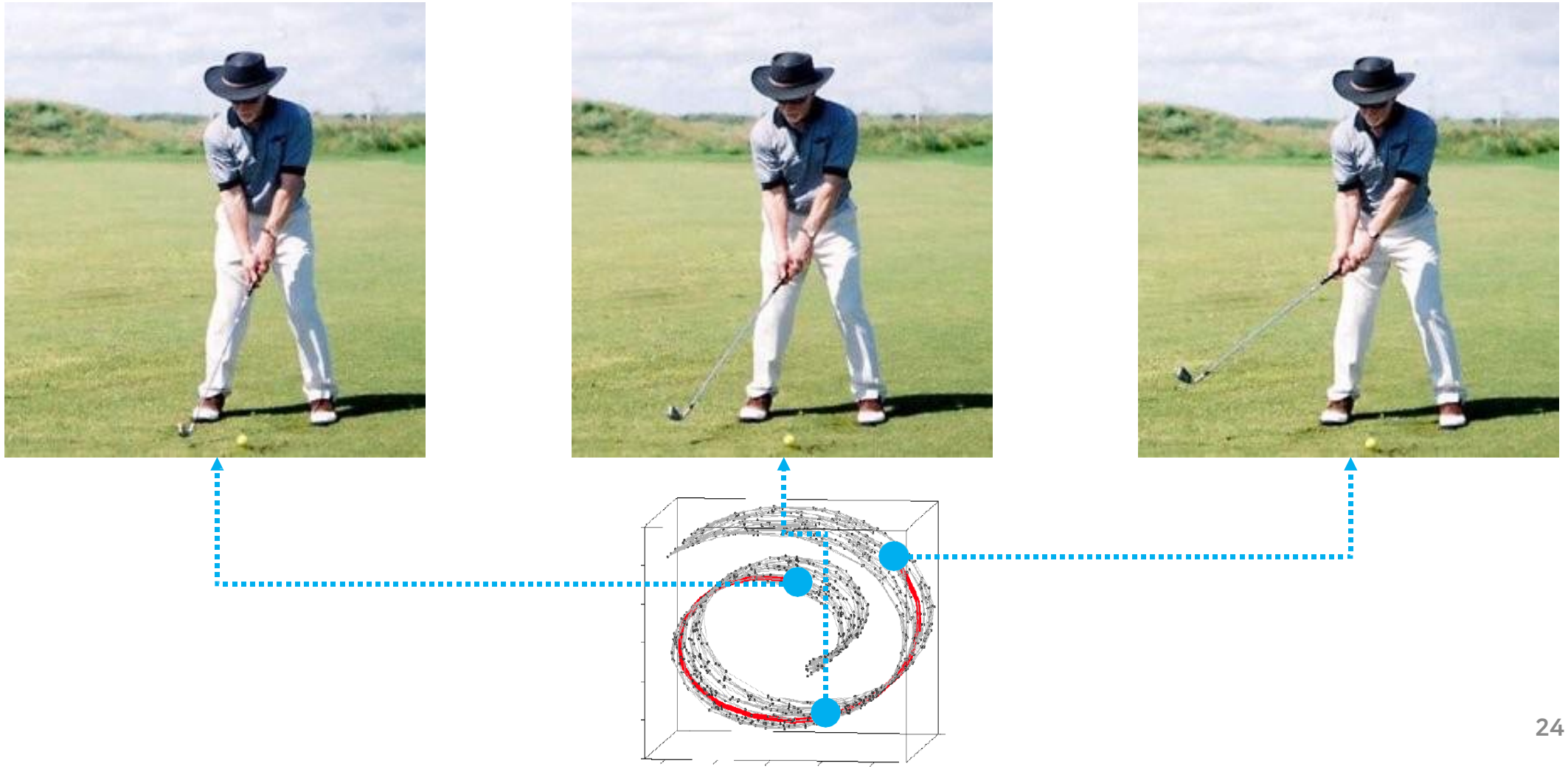
Distance in high dimension



2. Auto-Encoder(=A.E.)

사용목적 : Manifold-Learning; Encoder

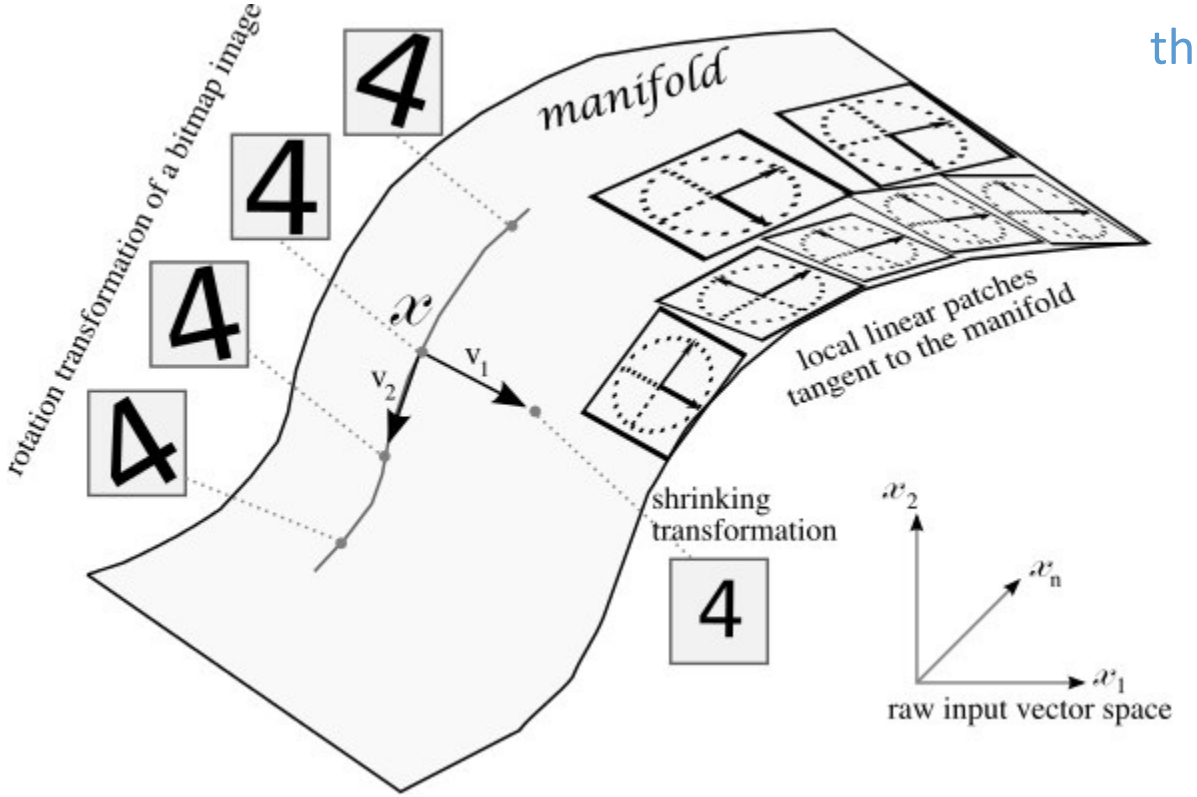
Distance in manifold



2. Auto-Encoder(=A.E.)

사용목적 : Manifold-Learning; Encoder

How can we understand Latent-Vector ?



From InfoGAN

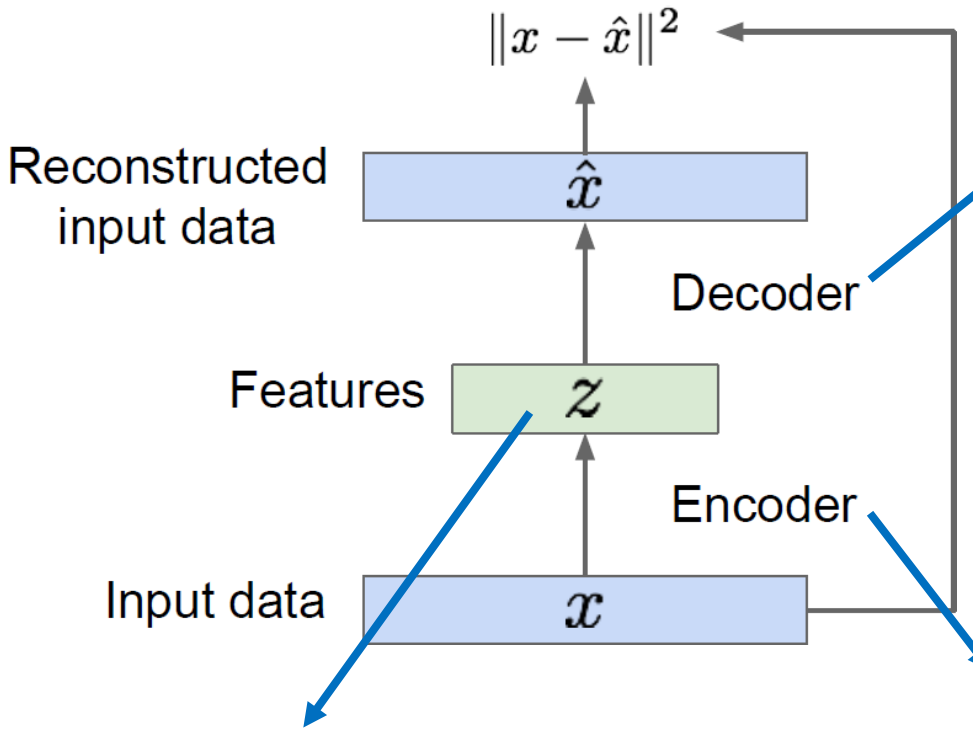
From VAE

2. Auto-Encoder(=A.E.)

사용목적 : Manifold-Learning; Encoder

reconstruction error $L(x, y)$

L2 Loss function:

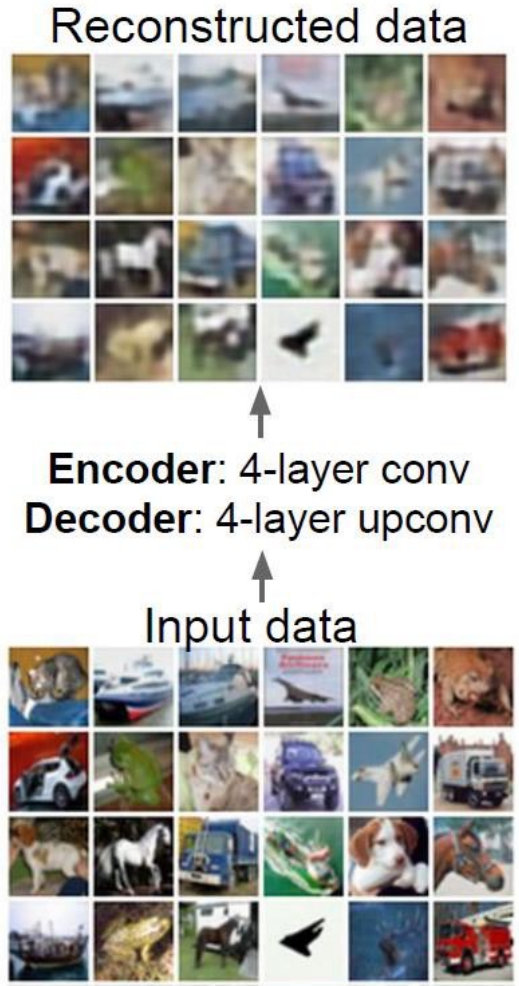


z usually smaller than **x** (dimensionality reduction)

Originally: Linear + no nlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN (upconv)

Originally: Linear + no nlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN

Doesn't use labels!

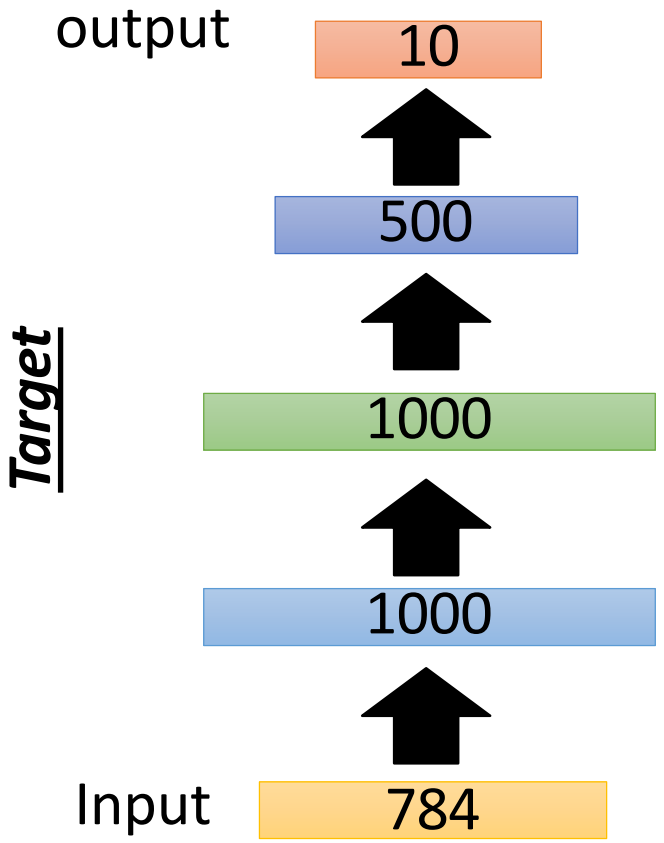


2. Auto-Encoder(=A.E.)

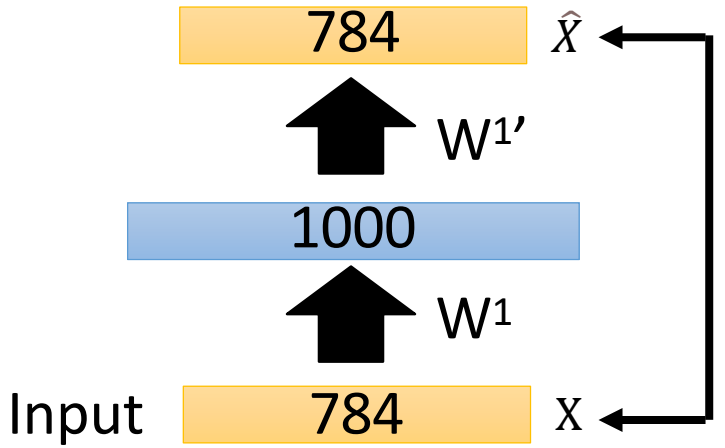
A.E.의 다른 활용법(1) : Stacking A.E.(=S.A.E.)

Stacking Autoencoder

-Transfer-Learning에서 가중치 초기화하는데 사용



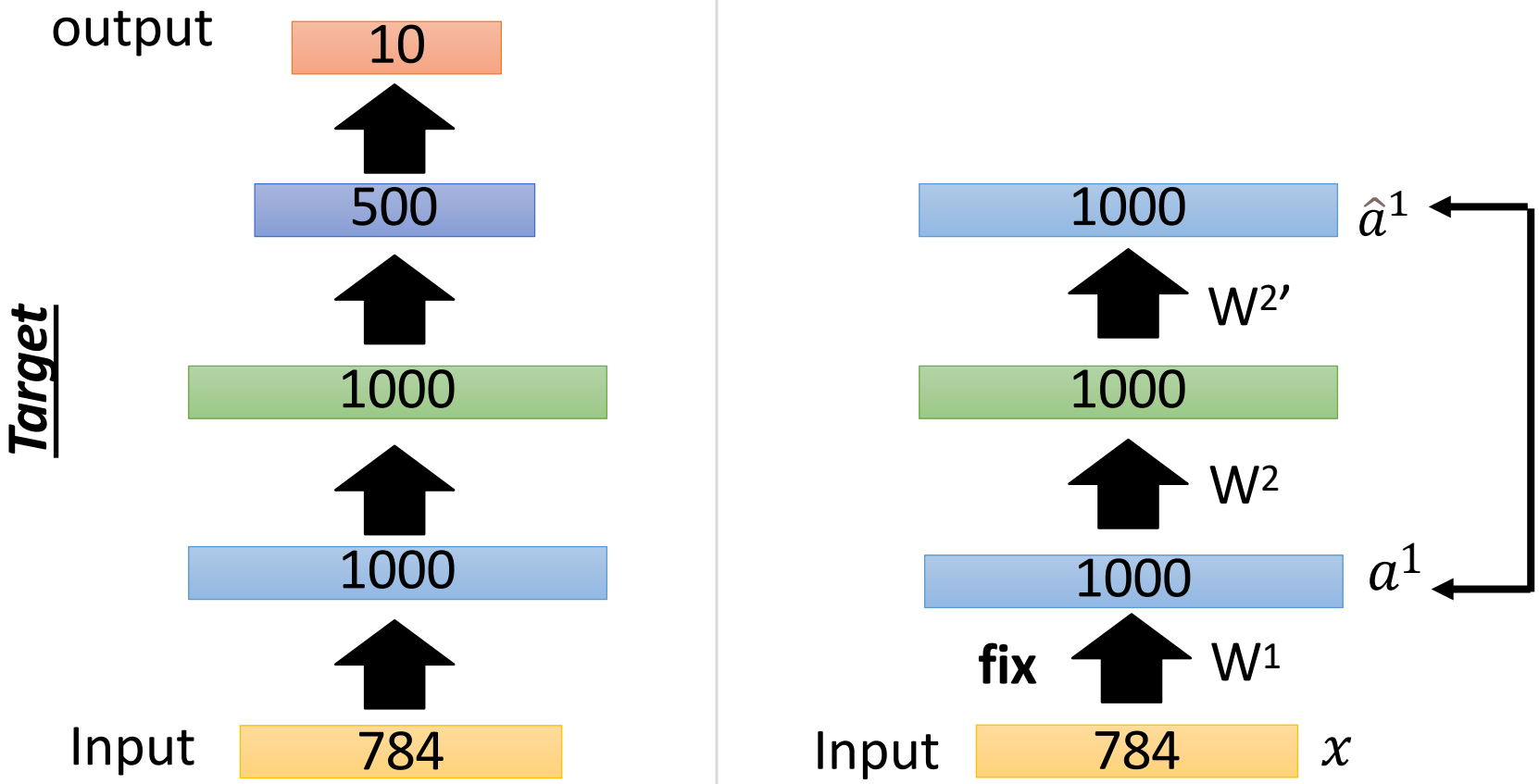
- 2000년대 초반의 A.E.는 전이학습(transfer-learning)을 위한 가중치 초기화(Weight-Initialization)를 하기 위해 사용되었다.
- 방법 : Input과 가까운 layer의 weights부터 차례대로 A.E.를 가정하여 가중치를 학습시킨다.
- 의미 : 입력값에 대해서 다시 복원해줄 수 있는 정보가 들어간 weights로 초기화를 진행시킨다.



2. Auto-Encoder(=A.E.)

A.E.의 다른 활용법(1) : Stacking A.E.(=S.A.E.)

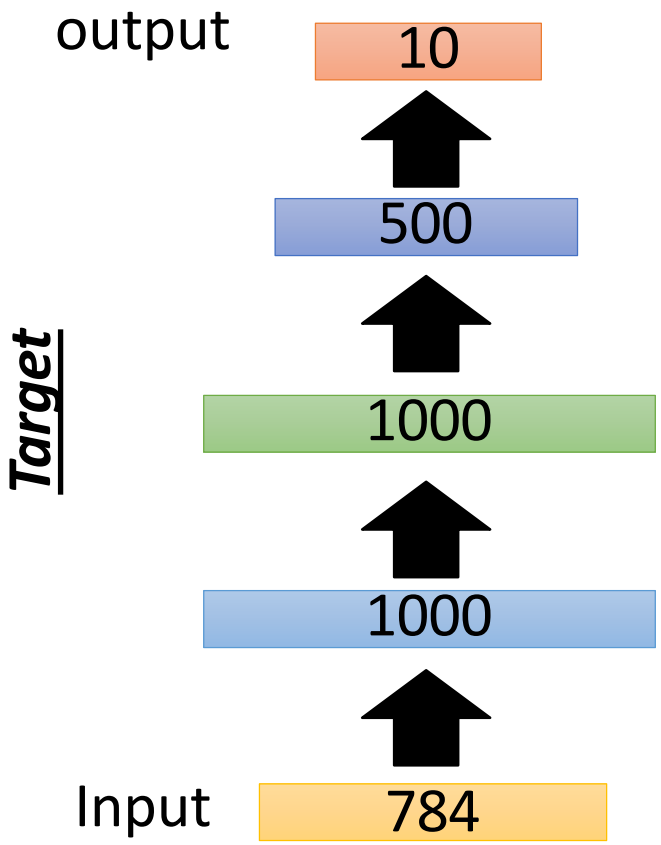
Stacking Autoencoder

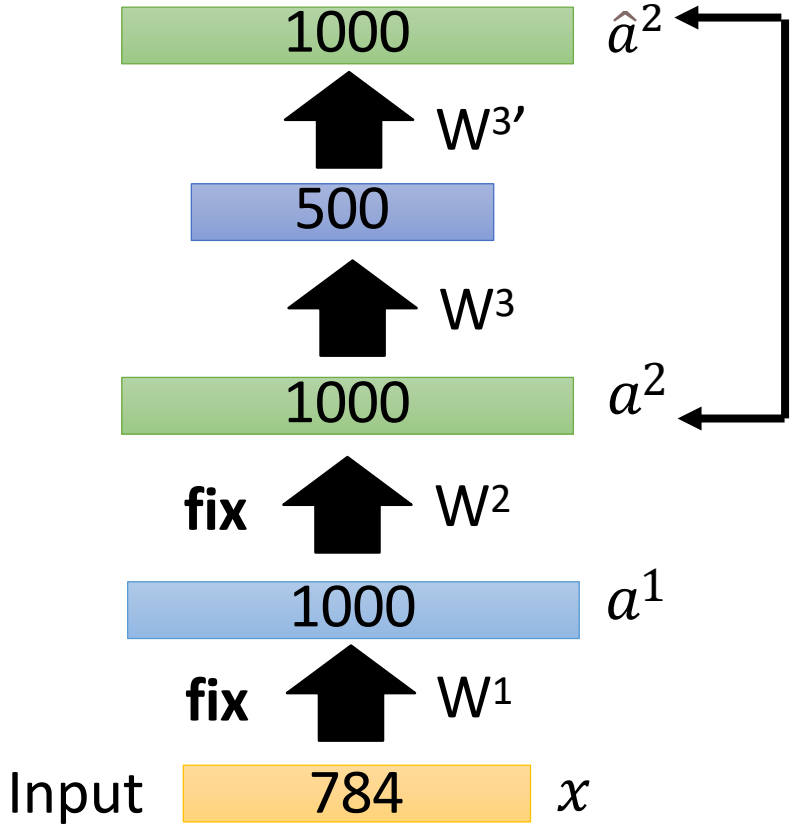


2. Auto-Encoder(=A.E.)

A.E.의 다른 활용법(1) : Stacking A.E.(=S.A.E.)

Stacking Autoencoder

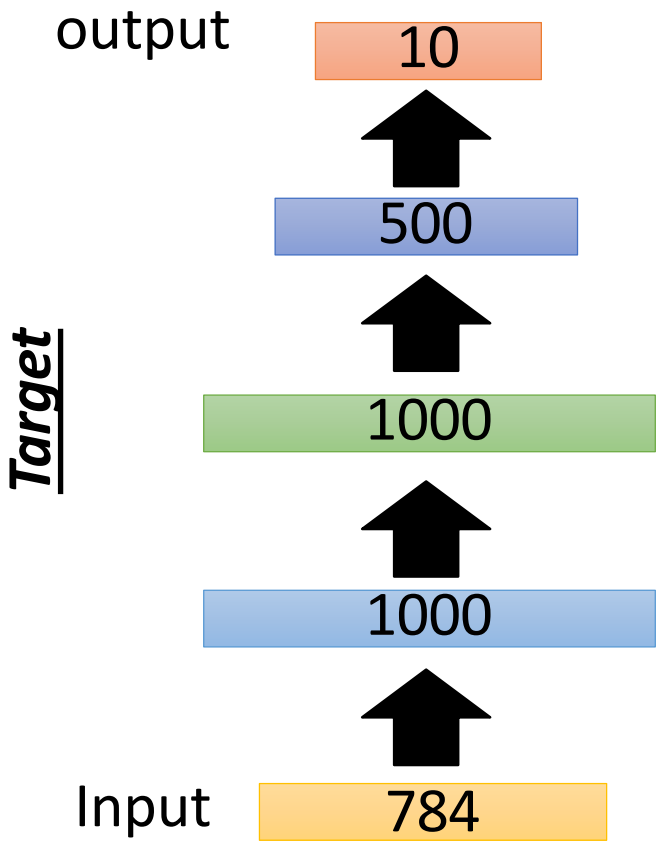




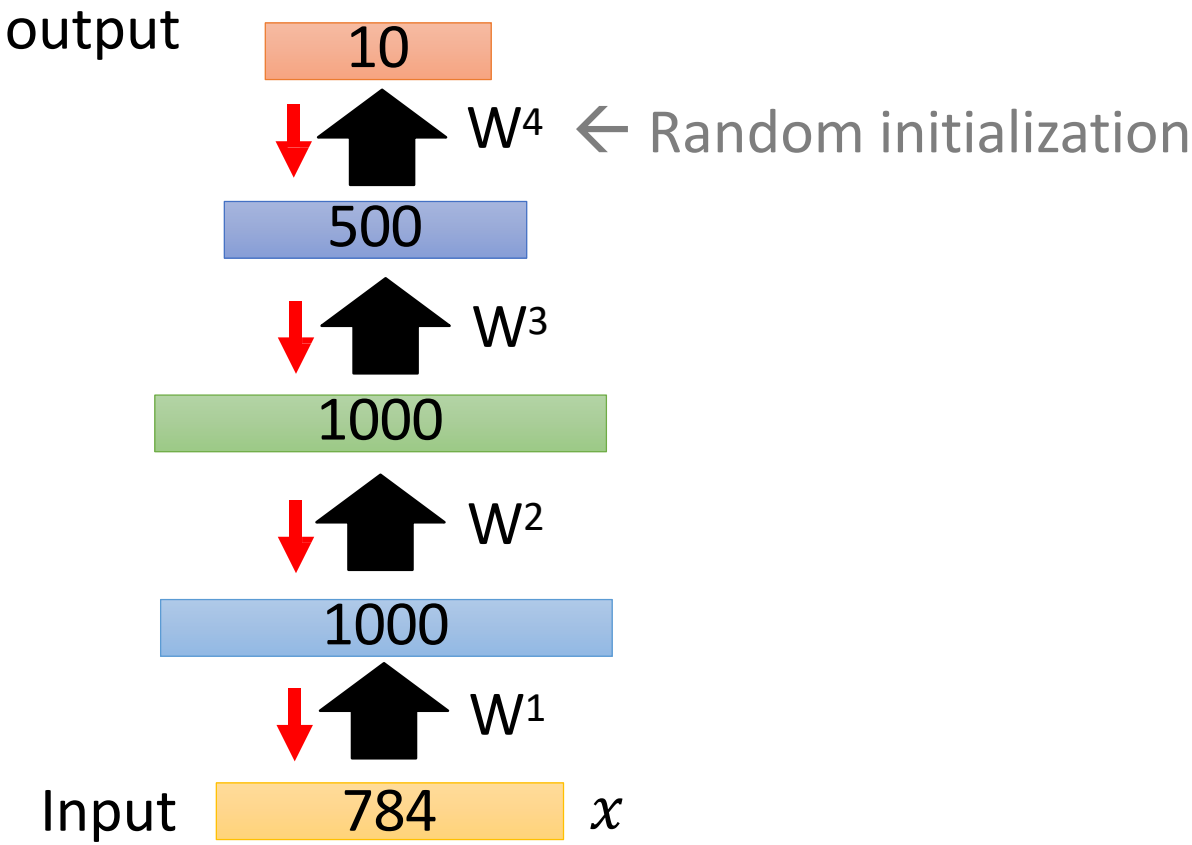
2. Auto-Encoder(=A.E.)

A.E.의 다른 활용법(1) : Stacking A.E.(=S.A.E.)

Stacking Autoencoder



Fine-tuning by backpropagation

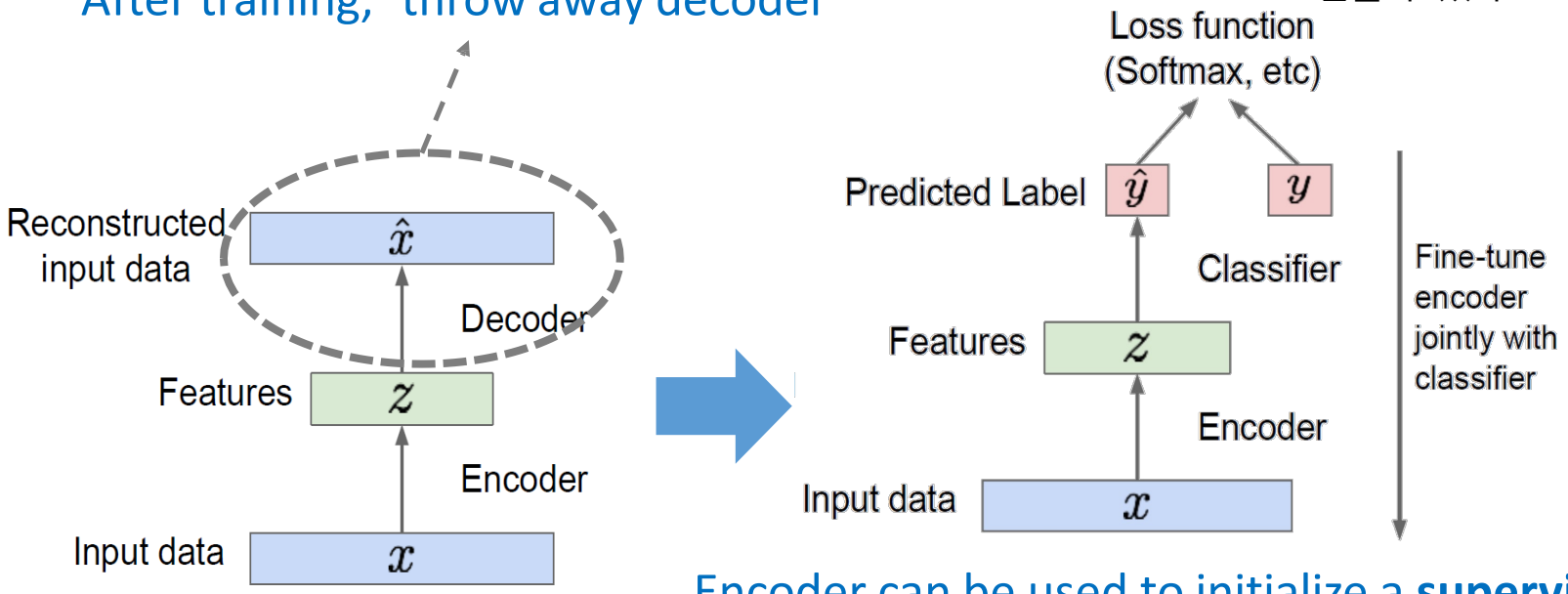


2. Auto-Encoder(=A.E.)

A.E.의 다른 활용법(2) : Supervised Learning

- 이전과 동일하게 전이학습(fine-tunning)에 사용된다. 그러나 그 방법에서 조금 차이를 갖는다.
- 이전과 달리 전체 N.N. 중에 features-extraction 부분을 Encoder로 하여 Decoder를 붙여서 A.E.를 만든 후에 모델을 학습시킨다.
- 학습시킨 이후, Decoder를 제거하고 대신에 classification를 수행하는 부분을 합쳐 classification-model을 만든다.
- Label된 데이터가 적은 경우에 해당 방법을 통해 classification-model을 만들 수 있다.

After training, throw away decoder

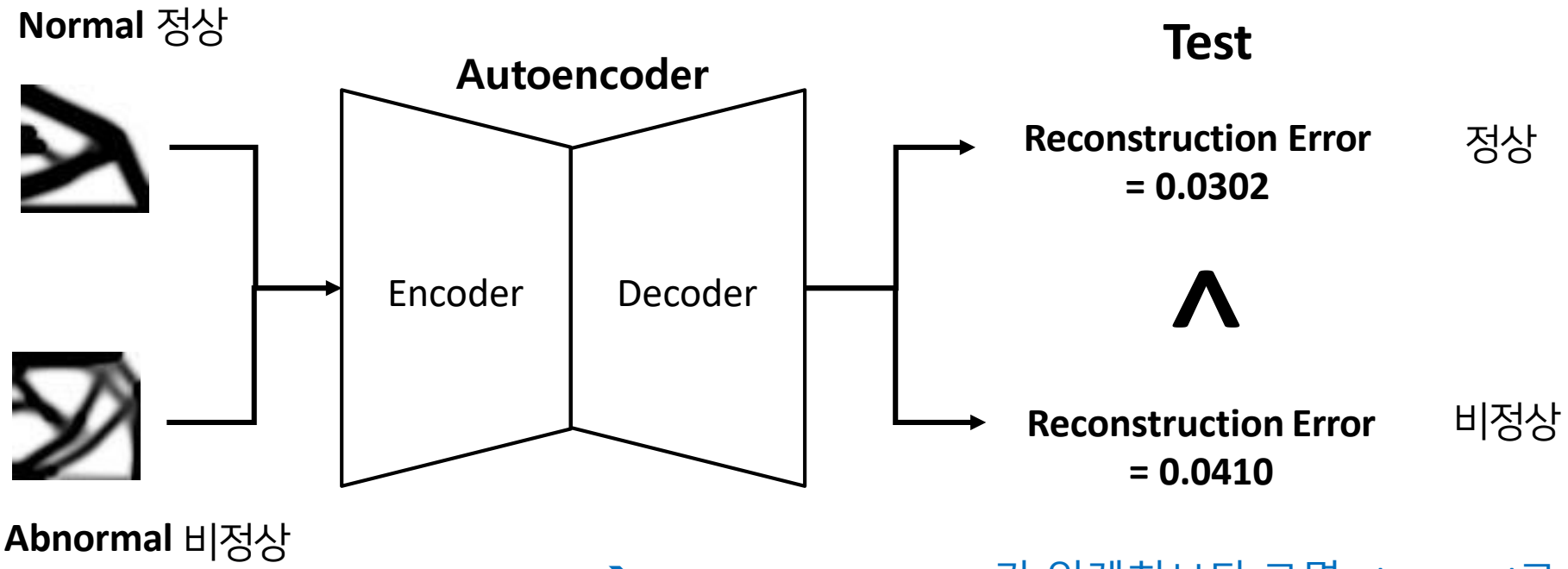


Encoder can be used to initialize a **supervised** model

2. Auto-Encoder(=A.E.)

A.E.의 다른 활용법(3) : Anomaly Detection(이상치 탐지)

- A.E.의 input-data인 X로 normal(정상) 데이터만 넣어서 학습시킨다.
- 후에 어떤 사진(X)을 넣었을 때, 만약 output값인 $\hat{X}$ 가 X와의 차이가 클수록 복원이 안된다는 뜻임으로 투입된 어떤 사진(X)은 abnormal(비정상)으로 분류한다.
- 보통적으로 정상데이터가 더 많음으로, 이런 방법이 용이하다.



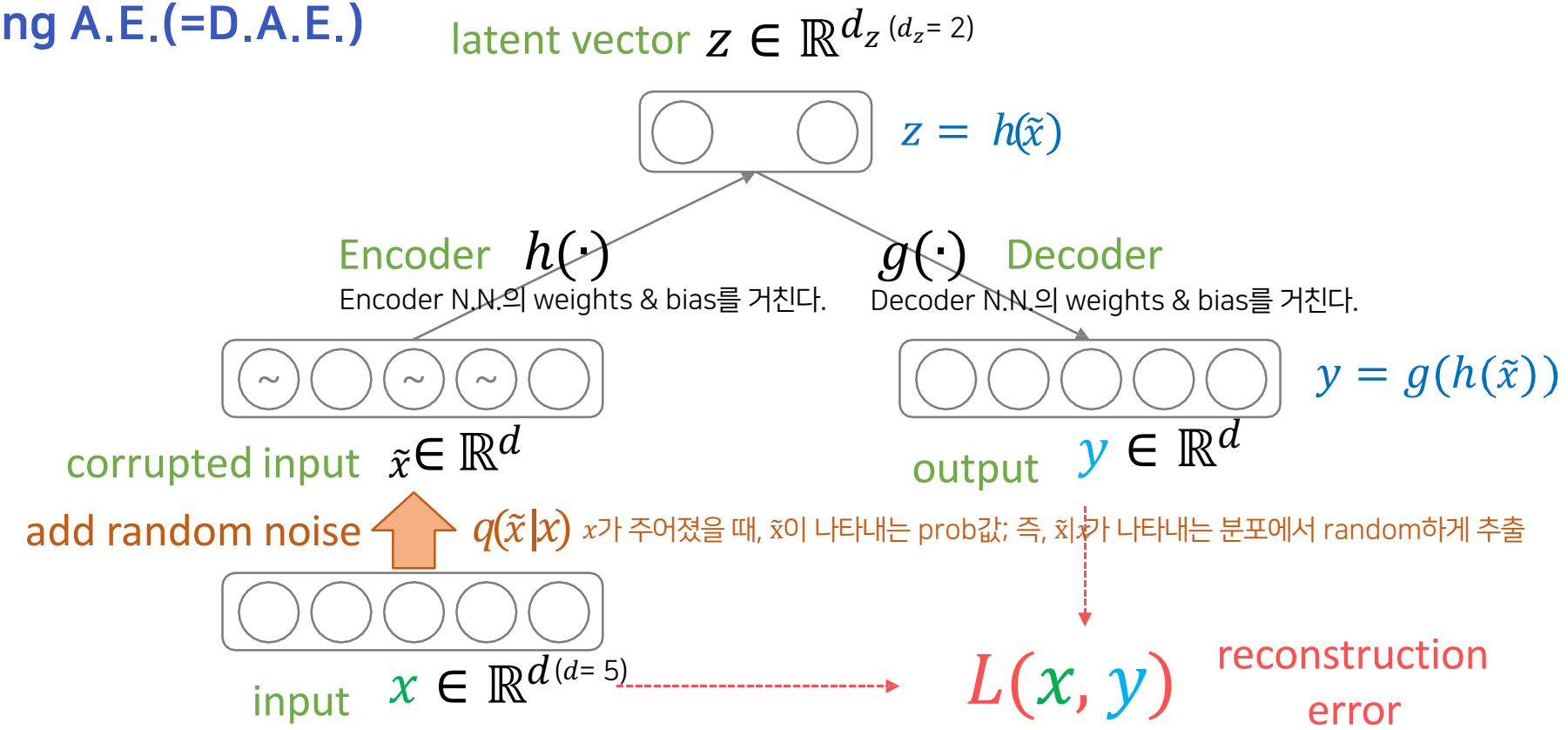
➔ Reconstruction Error가 임계치보다 크면 Abnormal로 분류

DATA
SCIENCE LAB

03. 여러가지 A.E.

3. 여러가지 A.E.

Denoising A.E.(=D.A.E.)



$random$한 각 $prob = q(\tilde{x}|x)$에 대하여, 각 $Loss$를 가중치 하여 더한다.

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

$x \in D$ 모든 데이터셋에 대해 Loss를 개별적으로 계산해서 더한다.

3. 여러가지 A.E.

Denoising A.E.(=D.A.E.)

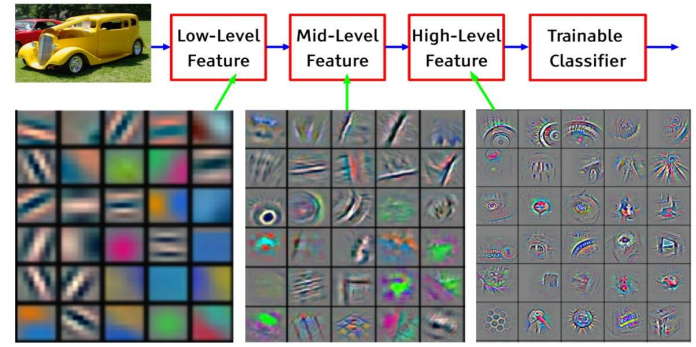
- Denoiseing : 'De' + 'Noise' : $\tilde{x}$ 에서 Noise를 없애다.
- Random Noise : 사람이 봤을 때, 구분가능한 최대 noise 추가
- Cannot compute expectation exactly, So, Use sampling corrupted inputs

$$L_{DAE} = \sum_{x \in D} E_{q(\tilde{x}|x)} \left[L\left(x, g\left(h(\tilde{x})\right)\right) \right] \approx \sum_{x \in D} \frac{1}{L} \sum_{i=1}^{L} L\left(x, g\left(h(\tilde{x})\right)\right)$$

L개 샘플에 대한 평균으로 대체

3. 여러가지 A.E.

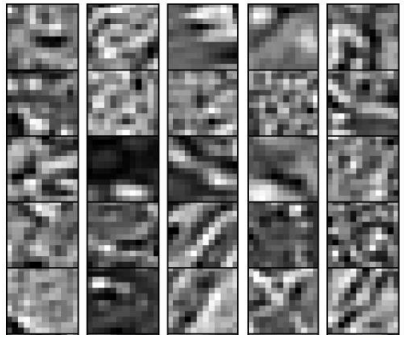
Denoising A.E.(=D.A.E.)



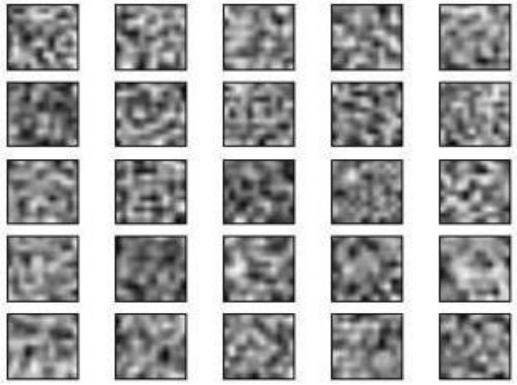
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Natural image patches (12x12 pixels) : 100 hidden units

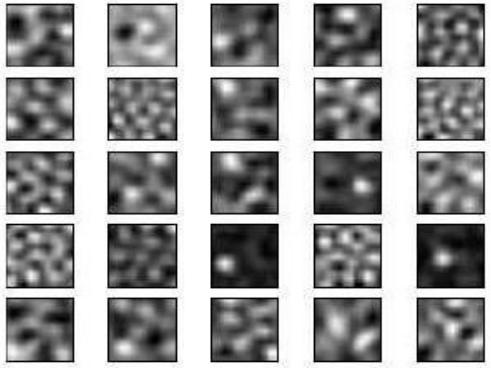
랜덤값으로 초기화하였기 때문에 노이즈처럼 보이는 필터일 수록 학습이 잘 안 된 것이고 edge filter와 같은 모습 일수록 학습이 잘 된 것이다.



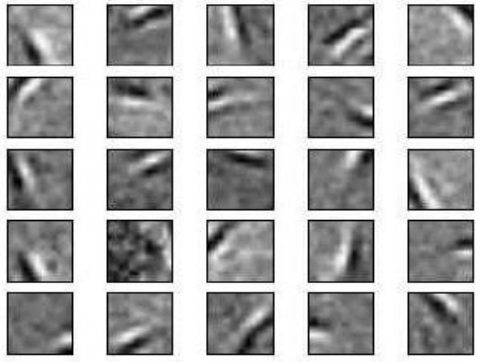
AE



AE with weight decay



DAE

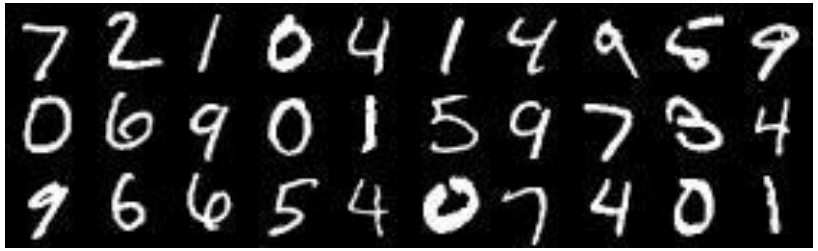


10% salt-and-pepper noise

3. 여러가지 A.E.

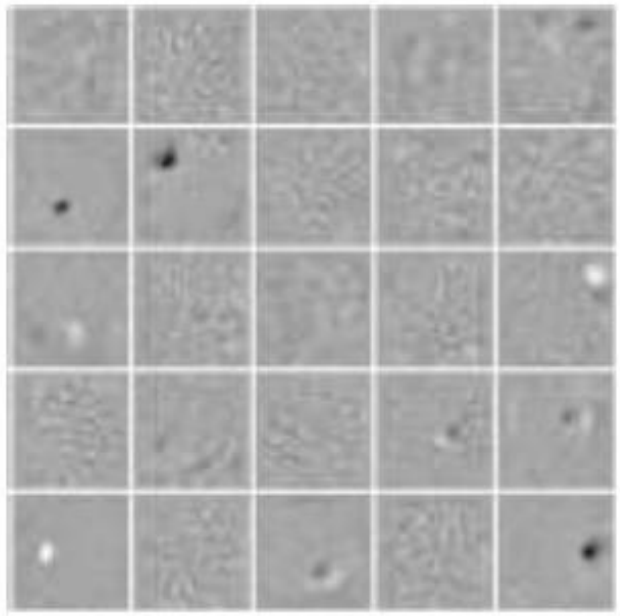
Denoising A.E.(=D.A.E.)

MNIST digits (64x64 pixels)

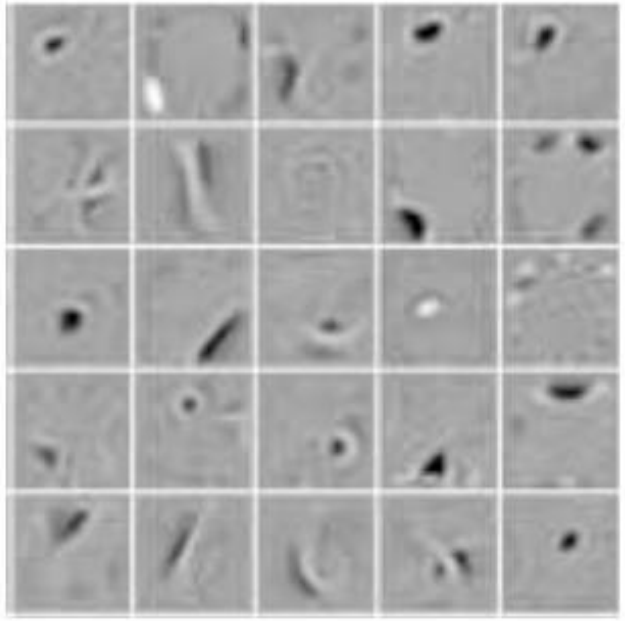


- Cross Entropy
- 100 hidden units
- Zero-masking noise

AE

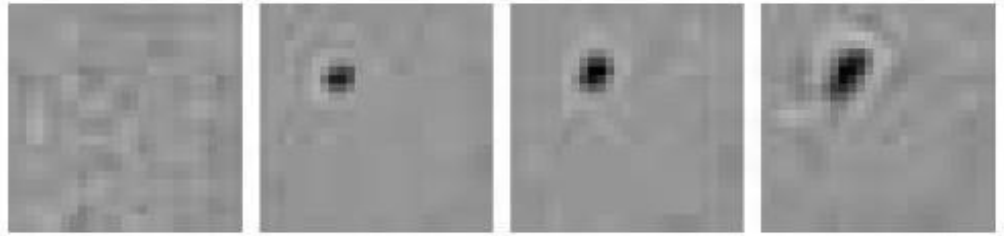


DAE



25% corruption

Increasing noise



(d) Neuron A (0%, 10%, 20%, 50% corruption)

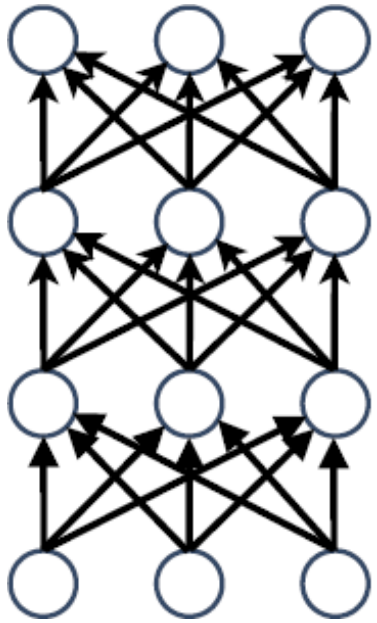
(e) Neuron B (0%, 10%, 20%, 50% corruption)

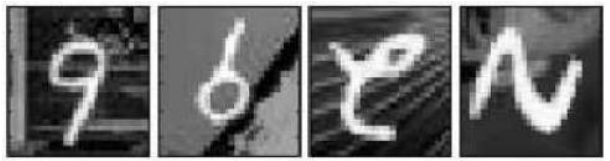
3. 여러가지 A.E.

Denoising A.E.(=D.A.E.)

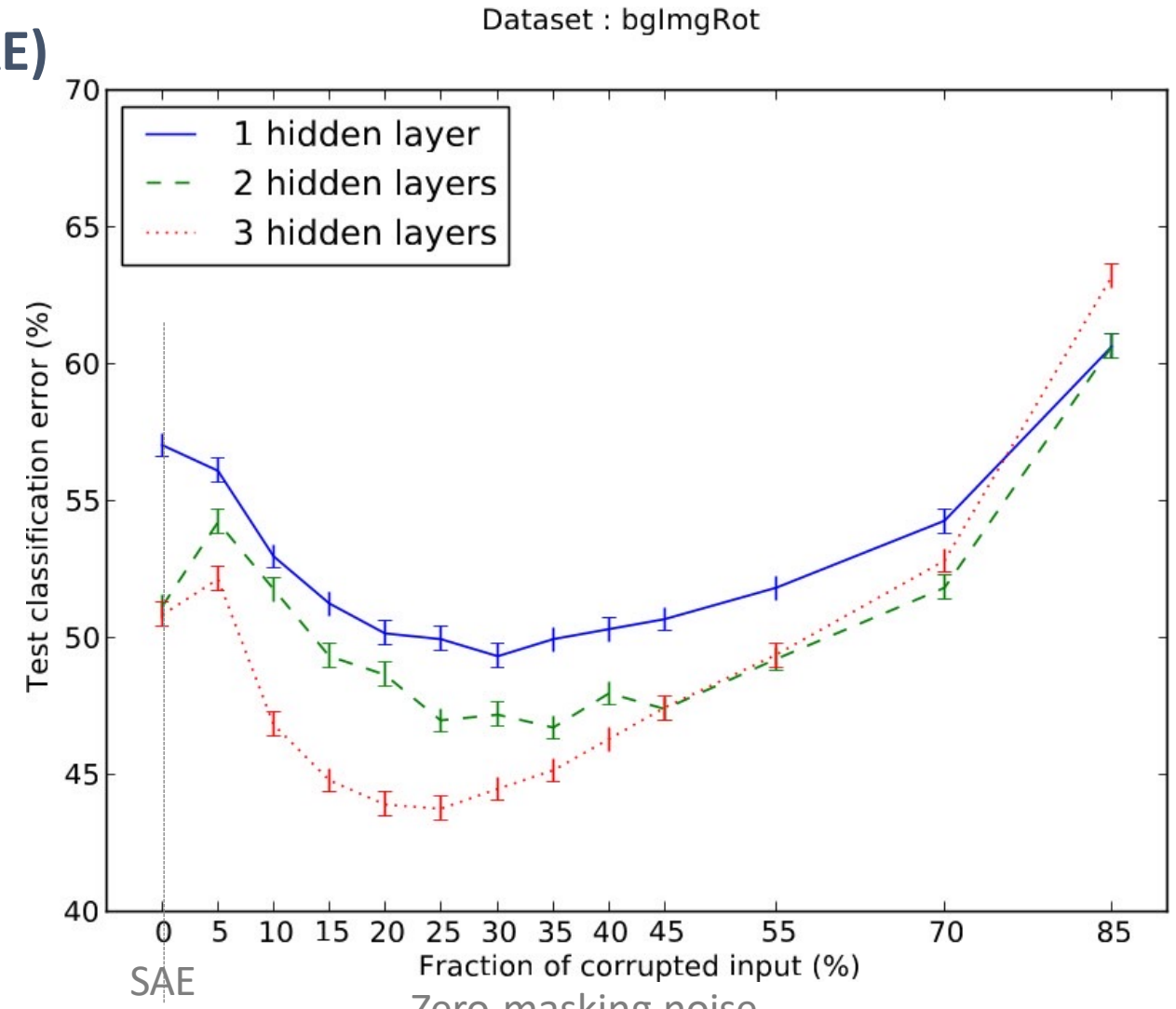
Stacked Denoising Auto-Encoders (SDAE)

Performance – Pretraining





bgImgRot Data Train/
Valid/Test : 10k/2k/20k



SAE

Zero-masking noise

Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion

3. 여러가지 A.E.

Contractive A.E.(=C.A.E.)

Stochastic Contractive AutoEncoder (SCAE)

DAE encourages reconstruction to be insensitive to input corruption

DAE의 loss의 해석은 g,h중에서 특히 h는 데이터 x가 조금 바뀌더라도 매니폴드 위에서 같은 샘플로 매칭이 되도록 학습 되어야 한다라고 볼 수 있다.

Alternative: encourage representation to be insensitive

추가된 항으로, $noise\ X$와 그냥 원본 X를 $Encoder\ N.N.$에 $input$으로 넣었을 때, $output$값인 $latent-vector$가 유사하도록 하는 규제항; $h(\ )$: Encoder-Function

$$L_{DAE} = \underbrace{\sum_{x \in D} L\left(x, g\left(h(\tilde{x})\right)\right)}_{\text{Reconstruction Error}} + \underbrace{\lambda E_{q(\tilde{x}|x)}\left[\left|\left|h(x) - h(\tilde{x})\right|\right|^2\right]}_{\text{Stochastic Regularization}}$$

($A.E.$의 $Loss-Function$항)

Tied weights i.e. $W' = W^T$ prevent W from collapsing h to 0.

정규화 항목이 0이 되는 경우는 h가 0인 경우도 있으므로 이를 방지하기 위해 tied weight를 사용한다.

3. 여러가지 A.E.

Contractive A.E.(=C.A.E.)

Contractive AutoEncoder (CAE)

SCAE stochastic regularization term : $E_{q(\tilde{x}|x)}\left[\left|\left|h(x)-h(\tilde{x})\right|\right|^2\right]$

For small additive noise, $\tilde{x}|x = x+\epsilon\ \epsilon\sim N(0,\sigma^2 I),$

Taylor series expansion yields, $h(\tilde{x}) = h(x+\epsilon) = h(x) + \frac{\partial h}{\partial x}\epsilon + \cdots$

It can be showed that

$$\underbrace{E_{q(\tilde{x}|x)}\left[\left|\left|h(x)-h(\tilde{x})\right|\right|^2\right]}_{\text{Stochastic Regularization (SCAE)}} \approx \underbrace{\left\|\frac{\partial h}{\partial x}(x)\right\|_F^2}_{\text{Analytic Regularization (CAE)}}$$

Frobenius Norm

$$||A||_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2$$

3. 여러가지 A.E.

Contractive A.E.(=C.A.E.)

$$\sum_{x \in D} \underbrace{L\left(x, g\left(h(\tilde{x})\right)\right)}_{\text{Reconstruction Error}} + \underbrace{\left\| \frac{\partial h}{\partial x}(x) \right\|_F^2}_{\text{Analytic Contractive Regularization}}$$

For training examples, encourages both:

- small reconstruction error
- representation insensitive to small variations around example

Analytic contractive regularization term is

- Frobenius norm of the Jacobian of the non-linear mapping

$$\left\| \frac{\partial h}{\partial x}(x) \right\|_F^2 = \sum_{ij} \left(\frac{\partial z_j}{\partial x_i}(x) \right)^2 = \| J_h(x) \|_F^2$$

Penalizing $\| J_h(x) \|_F^2$ encourages the mapping to the feature space to be contractive in the neighborhood of the training data.

highlights the advantages for representations to be locally invariant in many directions of change of the raw input.

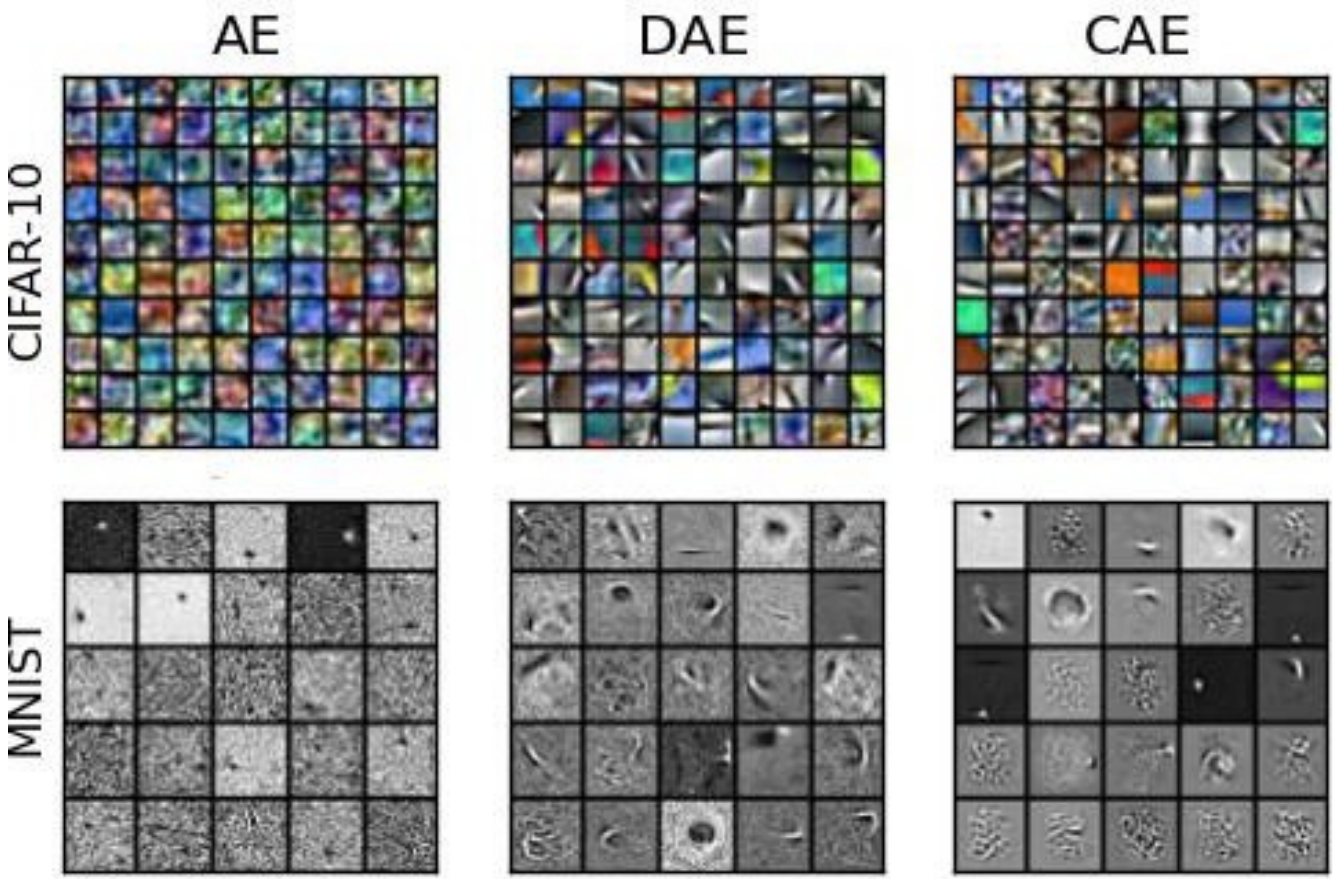
3. 여러가지 A.E.

Contractive A.E.(=C.A.E.)

- 4000 hidden units
- 2000 hidden units

Gaussian noise

CIFAR-10 (32x32 pixels)
MNIST digits (64x64 pixels)



DATA
SCIENCE LAB

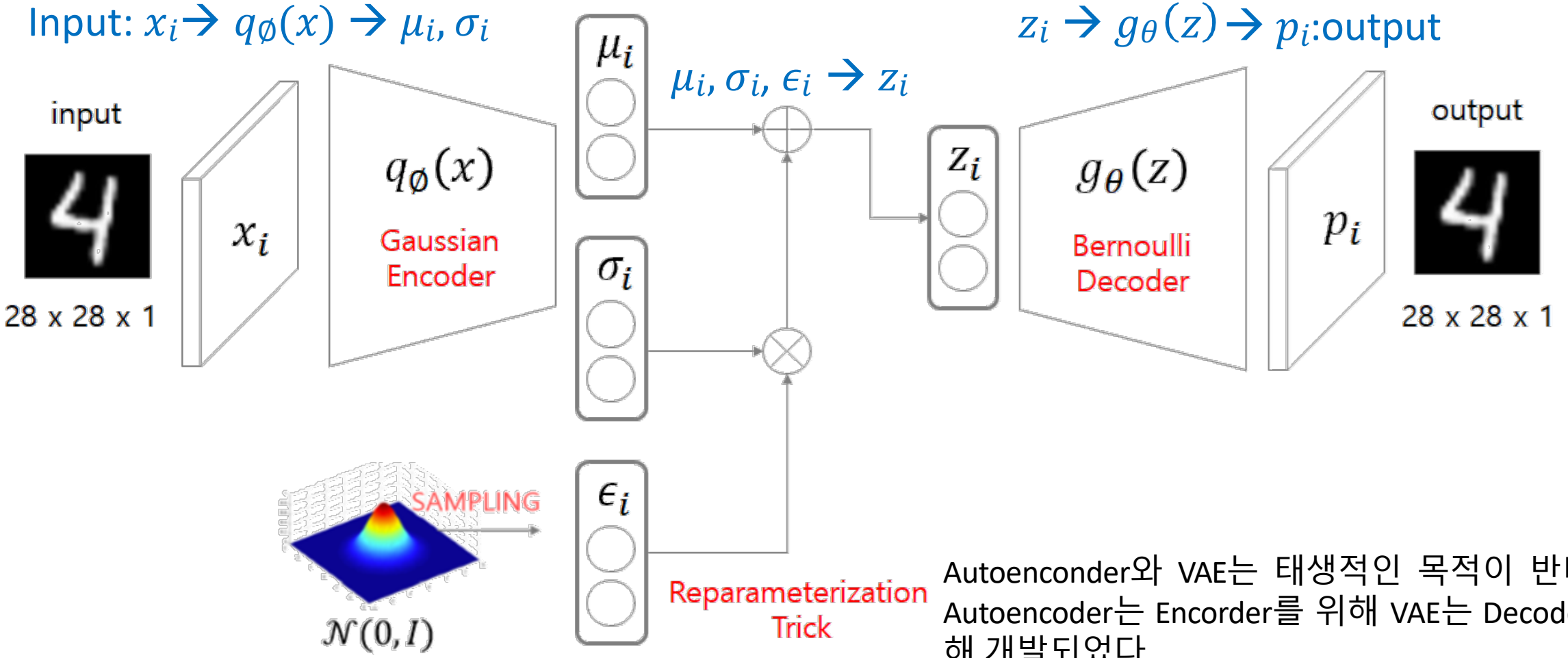
04. Variational Auto-Encoder(=V.A.E.)

4. Variational Auto-Encoder(=V.A.E.)

전체적인 구조

Input: $x_i \rightarrow q_\phi(x) \rightarrow \mu_i, \sigma_i$

$z_i \rightarrow g_\theta(z) \rightarrow p_i$:output



Autoenconder와 VAE는 태생적인 목적이 반대이다. Autoencoder는 Encorder를 위해 VAE는 Decoder를 위해 개발되었다.

4. Variational Auto-Encoder(=V.A.E.)

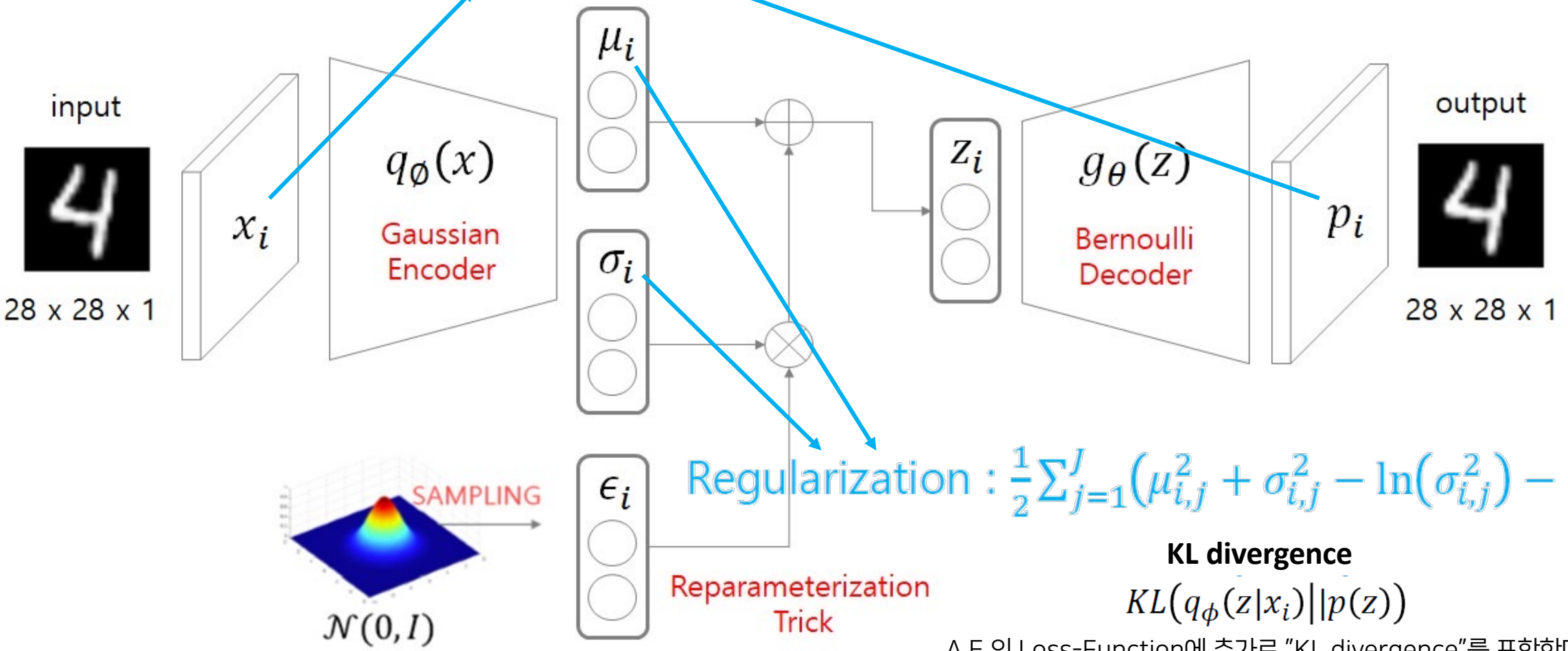
전체적인 구조

$$-\mathbb{E}_{q_\phi(z|x_i)}\left[\log\left(p(x_i|g_\theta(z))\right)\right]$$

Cross entropy

$$\text{Reconstruction Error: } -\sum_{j=1}^{D} x_{i,j}\log p_{i,j} + (1-x_{i,j})\log(1-p_{i,j})$$

A.E.의 Loss-Function이었던 "Reconstruction Error"를 포함한다.



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

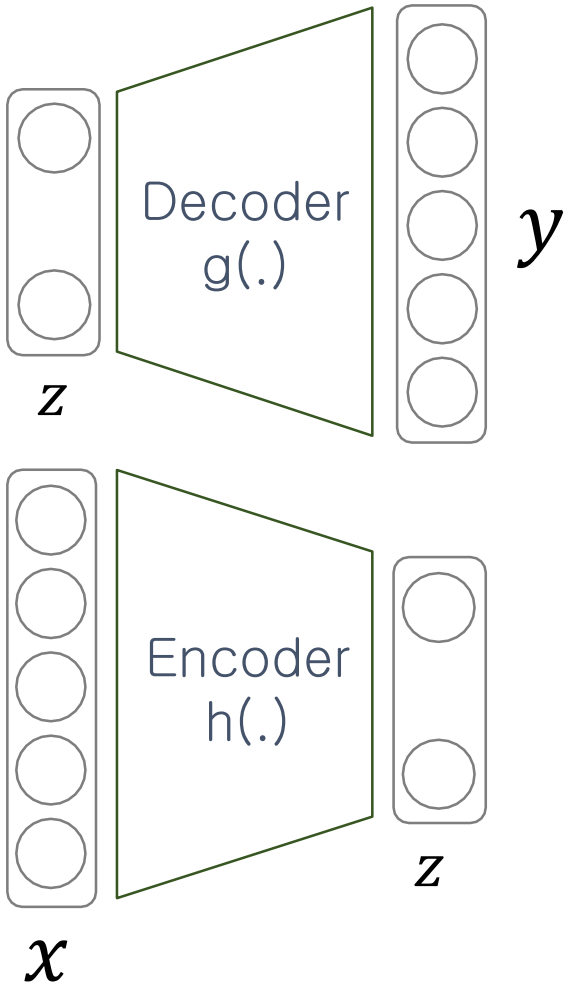
KL divergence

$$KL\left(q_\phi(z|x_i)\,||\,p(z)\right)$$

A.E.의 Loss-Function에 추가로 "KL divergence"를 포함한다. 참고로 KL divergence는 어떤 두 r.v.의 Distribution간의 유사도를 의미한다.

4. Variational Auto-Encoder(=V.A.E.)

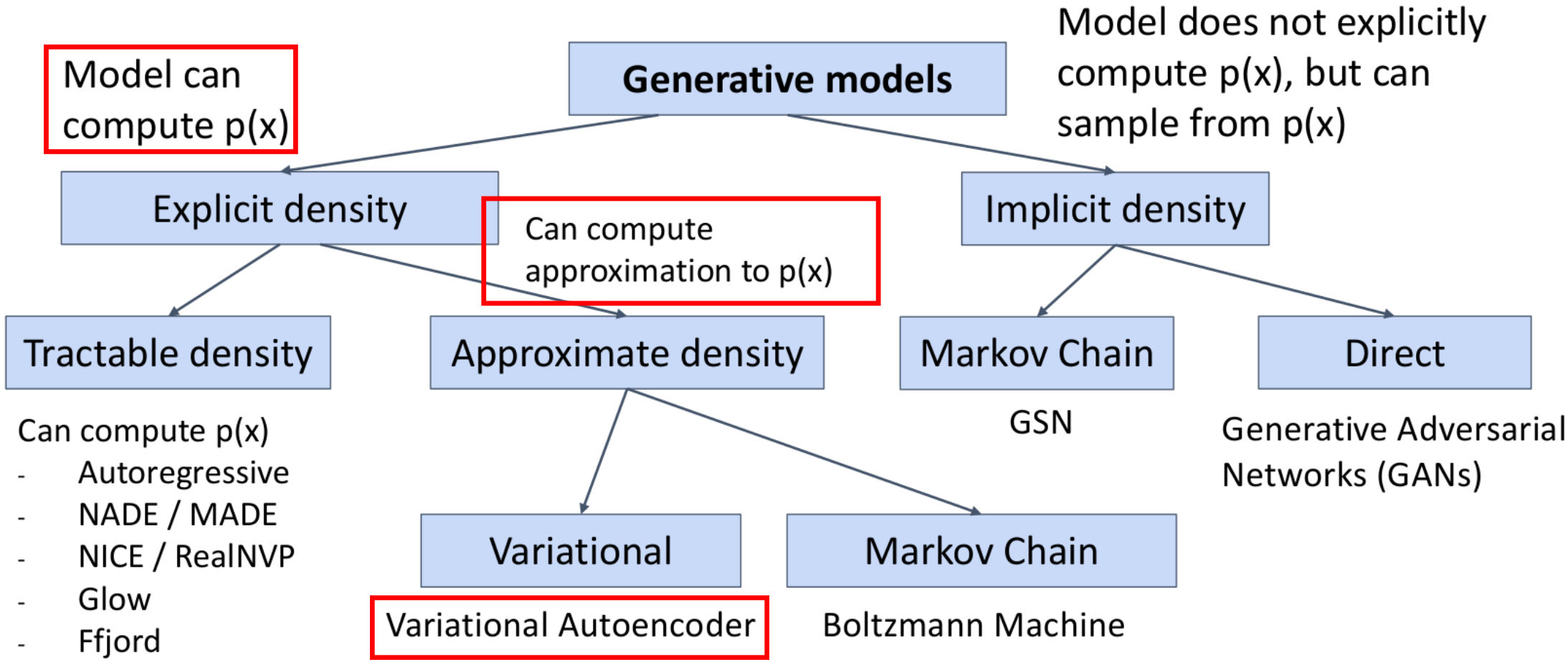
사용목적 : Generative-Model; Decoder



- Generative-Model 개념 : 어떤 특정한 벡터(Z; latent-vector)를 이용해 원하는 벡터 혹은 행렬 형태의 크기를 가진 데이터(y; output-data)를 만들고 싶다. 이 때, 어떤 특정한 벡터(Z)를 원하는 형태의 크기를 가진 데이터(y; output-data)로 확장시키는 N.N.을 Decoder라고 한다.
- 특정한 벡터(Z; latent-vector)는 어떻게 구할 것이며, 이들을 통해 어떻게 이미지를 생성해낼 수 있는가 ?
  - 특정한 벡터(Z; latent-vector)는 생성하고자 하는 이미지들의 표본데이터(X)를 입력데이터(X; input-data)로 하여, Encoder N.N.을 거친 값들로 한다. 의미적으로 보았을 때, 특정한 벡터(Z)는 Gaussian-Dist의 mean과 std에 대한 r.v.값들이 된다.
  - 이미지는 r.v.인 특정한 벡터(Z; latent-vector)으로 완성된 Distribution에서 sampling한 값을 이용해 만든다. 이에 대한 구체적인 과정은 이후에 다루도록 한다.

4. Variational Auto-Encoder(=V.A.E.)

사용목적 : Generative-Model; Decoder



4. Variational Auto-Encoder(=V.A.E.)

세부적인 구조(1). Loss-Function

Decoder의 구조와 r.v.

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from underlying unobserved (latent) representation **z**

Sample from true conditional

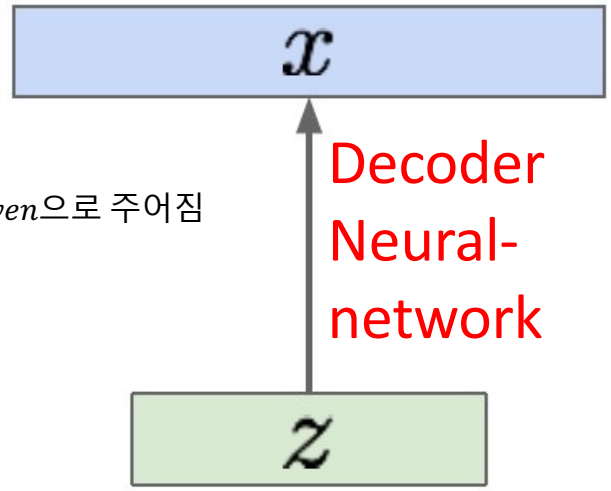
$p_{\theta^*}(x \mid z^{(i)})$

$x|z^{(i)}$은 $conditional - r.v.$이 때, $z^{(i)}$가 $given$으로 주어짐

Sample from true prior

$p_{\theta^*}(z)$

$z\ is\ r.v.\ \&\ prior$: 이미 알고 있다고 가정



Intuition (remember from autoencoders!): **x** is an image, **z** is latent factors used to generate **x:** attributes, orientation, etc.

4. Variational Auto-Encoder(=V.A.E.)

세부적인 구조(1). Loss-Function

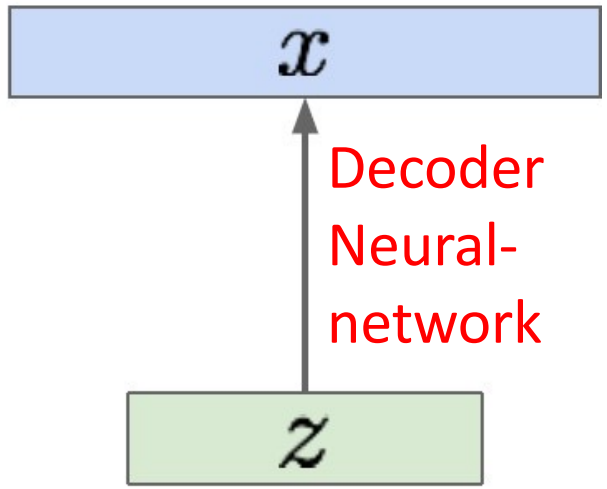
Decoder의 구조와 r.v.

We want to estimate the true parameters θ^* of this generative model.

How should we represent this model?

Sample from true conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$



Choose prior $p(z)$ to be simple, e.g. Gaussian or Bernoulli.

Conditional $p(x|z)$ is complex (generates image) => represent with neural network

4. Variational Auto-Encoder(=V.A.E.)

세부적인 구조(1). Loss-Function

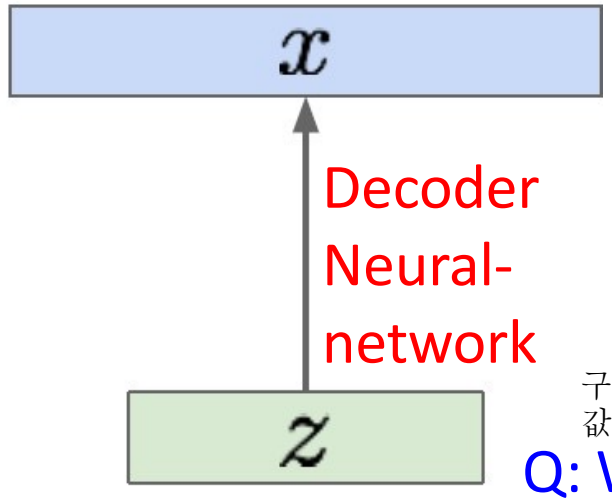
우리의 목표인 likelihoo값

We want to estimate the true parameters θ^* of this generative model.

How to train the model?

Sample from true conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$



Learn model parameters to maximize likelihood of training data

알고있다고 생각하는 값으로 가정을 통해 어떤 분포인지 결정할 수 있다.

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

구하고자 하는 값(*likelihood*)이자 최대화시키고자 하는 값이다.

Now with latent z

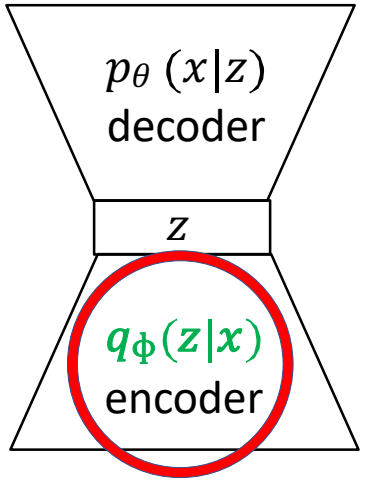
Q: What is the problem with this?

Intractable!

4. Variational Auto-Encoder(=V.A.E.)

세부적인 구조(1). Loss-Function

우리의 목표인 likelihoo값을 어떻게 구하는가 ?



1. Bayesian-Rule & 적분을 이용해 구하기

알고있다고 생각하는 값으로 가정을 통해 어떤 분포인지 결정할 수 있다.

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

구하고자 하는 값(*likelihood*)이자 최대화시키고자 하는 값이다.

Intractable to compute $p(x|z)$ for every z!

Gaussian or Benoulli Dist prior

Decoder neural network

모든 z에 대해$p(x|z)$을 계산하는데에 오랜 시간이 소요됨으로 이 방법은 포기

2. Bayesian-Rule을 이용해 구하기

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

Posterior density also intractable: Intractable data likelihood

Solution: In addition to decoder network modeling $p_\theta(x|z)$, define additional encoder network $q_\phi(z|x)$ that approximates $p_\theta(z|x)$

Will see that this allows us to derive a lower bound on the data likelihood that is tractable, which we can optimize.

4. Variational Auto-Encoder(=V.A.E.)

세부적인 구조(1). Loss-Function

Encoder를 도입한 V.A.E. 모델을 이용하여, log(likelihood)를 계산해보자.

$\left(p_\theta\left(x^{(i)}\right)\ Does\ not\ depend\ on\ z\right)$

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

Taking expectation wrt. z (using encoder network) will come in handy later

(Bayes' Rule)

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

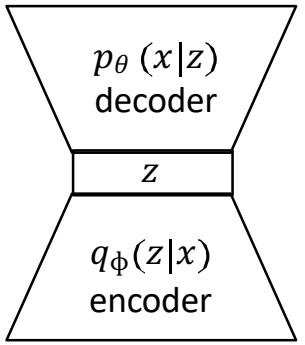
$$= \mathbf{E}_z\left[log\frac{p_\theta\left(x^{(i)}|z\right)p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)}\right]$$

$$= \mathbf{E}_z\left[\log\frac{p_\theta\left(\mathrm{x}^{(\mathrm{i})}|\mathrm{z}\right)\mathrm{p}_\theta(\mathrm{z})}{p_\theta\left(\mathrm{z}|\mathrm{x}^{(\mathrm{i})}\right)}\frac{q_\phi\left(\mathrm{z}|\mathrm{x}^{(\mathrm{i})}\right)}{q_\phi\left(\mathrm{z}|\mathrm{x}^{(\mathrm{i})}\right)}\right]$$ (Multiply by constant)

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

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

The expectation wrt. z (using encoder network) let us write nice KL terms



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

$$KL(P||Q) = \sum_x P(x)\log\frac{P(x)}{Q(x)}$$

4. Variational Auto-Encoder(=V.A.E.)

세부적인 구조(1). Loss-Function

Encoder를 도입한 V.A.E. 모델을 이용하여, log(likelihood)를 계산해보자.

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

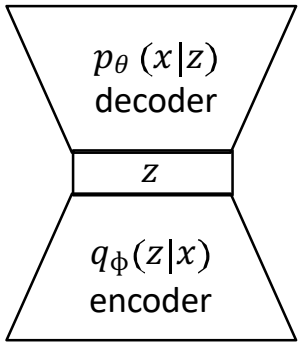
We want to maximize t he data likelihood

$$= \mathbf{E}_z\left[log\frac{p_\theta\left(x^{(i)}|z\right)p_\theta(z)}{p_\theta(z|x^{(i)})}\right]$$

$$= \mathbf{E}_z\left[\log\frac{p_\theta\left(\mathrm{x}^{(\mathrm{i})}|\mathrm{z}\right)\mathrm{p}_\theta(\mathrm{z})}{p_\theta\left(\mathrm{z}|\mathrm{x}^{(\mathrm{i})}\right)}\frac{q_\phi\left(\mathrm{z}|\mathrm{x}^{(\mathrm{i})}\right)}{q_\phi\left(\mathrm{z}|\mathrm{x}^{(\mathrm{i})}\right)}\right]$$ (Multiply by constant)

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

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



Decoder network gives $p_\theta(x|z)$, can compute estimate of t his term through sampling. (Sampling differentiable throug h reparam. trick, see paper.)

This KL term (between Gaussians for enco der and z prior) has nice closed-form soluti on!

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

4. Variational Auto-Encoder(=V.A.E.)

세부적인 구조(1). Loss-Function

Encoder를 도입한 V.A.E. 모델을 이용하여, log(likelihood)를 계산해보자.

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

We want to maximize t
he data likelihood

$$= \mathbf{E}_z\left[log\frac{p_\theta\left(x^{(i)}|z\right)p_\theta(z)}{p_\theta(z|x^{(i)})}\right]$$

$$= \mathbf{E}_z\left[\log\frac{p_\theta\left(\mathrm{x}^{(\mathrm{i})}|\mathrm{z}\right)\mathrm{p}_\theta(\mathrm{z})}{p_\theta\left(\mathrm{z}|\mathrm{x}^{(\mathrm{i})}\right)}\frac{q_\phi\left(\mathrm{z}|\mathrm{x}^{(\mathrm{i})}\right)}{q_\phi\left(\mathrm{z}|\mathrm{x}^{(\mathrm{i})}\right)}\right]$$ (Multiply by constant)

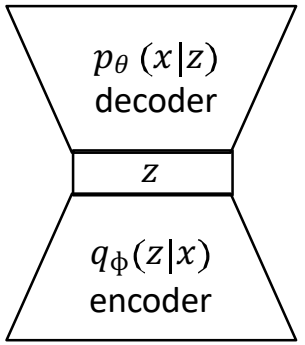
$$= \mathbf{E_z}\left[\log p_\theta\left(x^{(i)}|z\right)\right] - \mathbf{E_z}\left[log\frac{q_\phi\left(z|x^{(i)}\right)}{p_\theta(z)}\right] + \mathbf{E}_z\left[log\frac{q_\phi\left(z|x^{(i)}\right)}{p_\theta\left(z|x^{(i)}\right)}\right]$$ (Logarithms)

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

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

Tractable lower bound which we can take gradient of and optimize!

≥ 0



4. Variational Auto-Encoder(=V.A.E.)

세부적인 구조(1). Loss-Function

log(likelihood)를 Maximize하는 Loss-Function, 그리고 이에 대한 Lower-Bound값인 "ELBO"

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

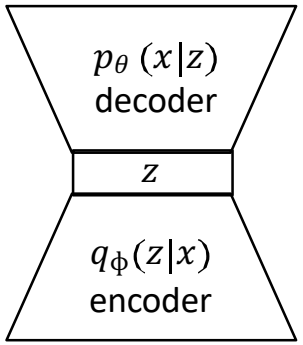
We want to maximize t
he data likelihood

$$= \mathbf{E}_z\left[log\frac{p_\theta\left(x^{(i)}|z\right)p_\theta(z)}{p_\theta(z|x^{(i)})}\right]$$

$$= \mathbf{E}_z\left[\log\frac{p_\theta\left(\mathrm{x}^{(\mathrm{i})}|\mathrm{z}\right)\mathrm{p}_\theta(\mathrm{z})}{p_\theta\left(\mathrm{z}|\mathrm{x}^{(\mathrm{i})}\right)}\ \frac{q_\phi\left(\mathrm{z}|\mathrm{x}^{(\mathrm{i})}\right)}{q_\phi\left(\mathrm{z}|\mathrm{x}^{(\mathrm{i})}\right)}\right]$$ (Multiply by constant)

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

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|\ z\right)\right] - D_{kL}\left(q_\phi\left(z|x^{(i)}\right)||p_\theta(z)\right)}_{\mathcal{L}(x^{(i)},\theta,\phi)} + \underbrace{D_{kL}\left(q_\phi\left(z|x^{(i)}\right)||p_\theta\left(z|x^{(i)}\right)\right)}_{\geq 0}$$



$$log p_\theta\left(x^{(i)}\right) \geq L\left(x^{(i)},\theta,\phi\right) \Rightarrow \theta^*,\phi^* = argmax_{\theta,\phi}\sum_{i=1}^{N} L\left(x^{(i)},\theta,\phi\right)$$

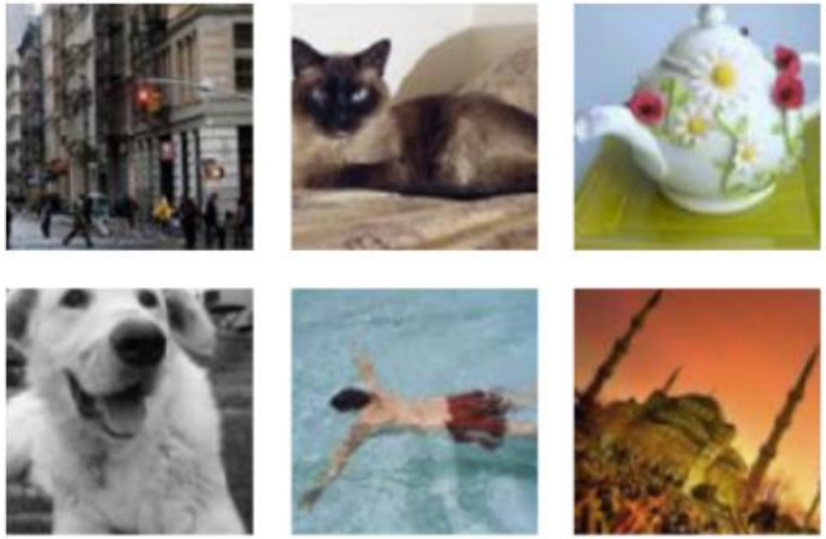
Variational lower bound ("ELBO")

Training: Maximize lower bound

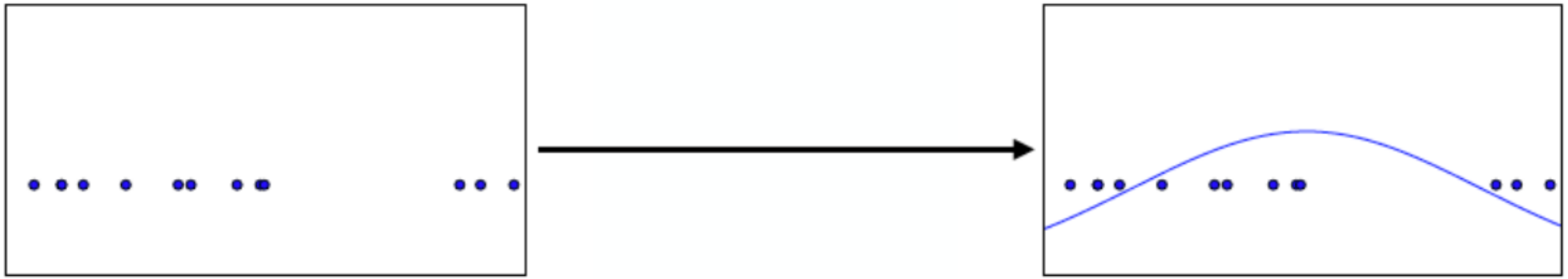
GENERATIVE MODEL

Sample Generation

Training Examples



Sampling

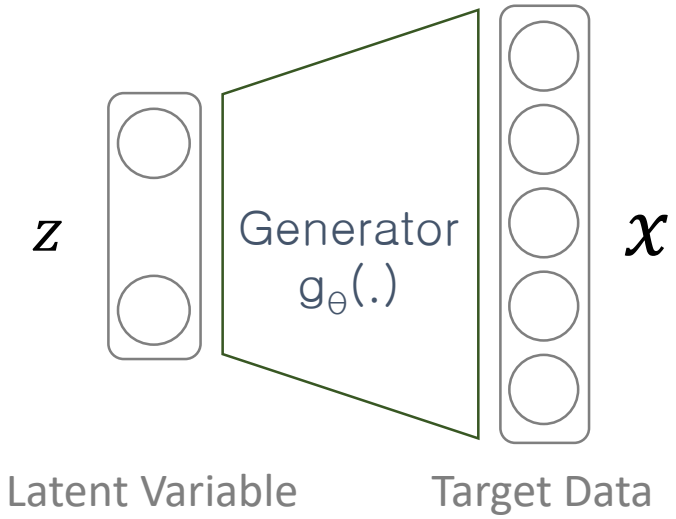


Density Estimation

https://github.com/mingyuliutw/cvpr2017_gan_tutorial/blob/master/gan_tutorial.pdf

GENERATIVE MODEL

Latent Variable Model



$z \sim p(z)$ Random variable

$g_\theta(\cdot)$ Deterministic function parameterized by θ

$x = g_\theta(z)$ Random variable

Latent variable can be seen as a set of control parameters for target data (generated data)

For MNIST example, our model can be trained to generate image which match a digit value z randomly sampled from the set [0, ..., 9].

그래서, p(z)는 샘플링 하기 용이해야 편하다.

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

We are aiming maximize the probability of each x in the training set, under the entire generative process, according to:

$$\int p(x|g_\theta(z))p(z)dz = p(x)$$

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 statitics. 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 레이어들를 통해 복잡할 수 있지만 딱 맞는 latent space로의 맵핑이 수행되고 나머지 레이어들을 통해 latent vector에 맞는 이미지를 생성할 수 있다.

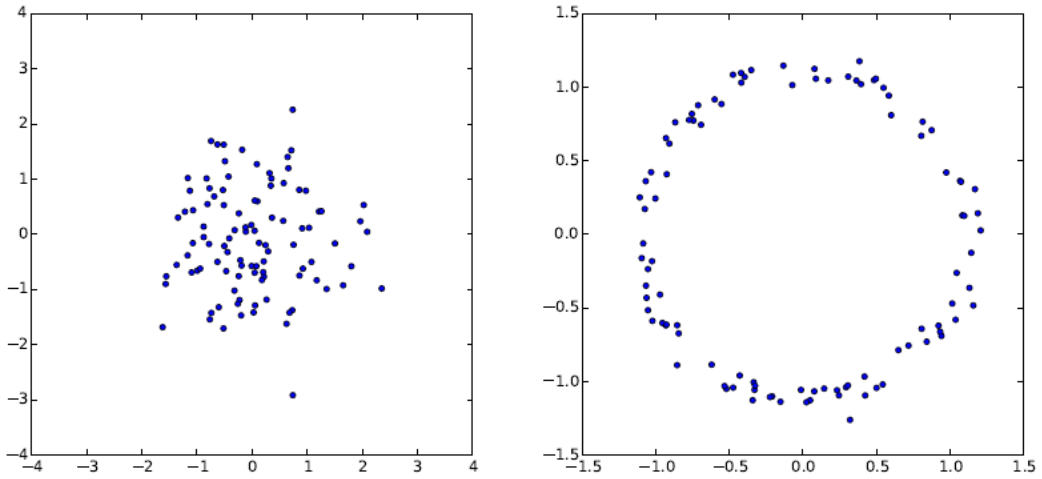


Figure 2: Given a random variable z with one distribution, we can create another random variable $X = g(z)$ with a completely different distribution. Left: samples from a gaussian distribution. Right: those same samples mapped through the function $g(z) = z/10 + z/||z||$ to form a ring. This is the strategy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

Tutorial on Variational Autoencoders : https://arxiv.org/pdf/1606.05908

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 σ hyperparameter of our Gaussian distribution such that this kind of erroroneous digit does not contribute to p(X) → hard..
Solution 2: we would likely need to sample many thousands of digits from z_{good} → hard..


(a) (b) (c)

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.

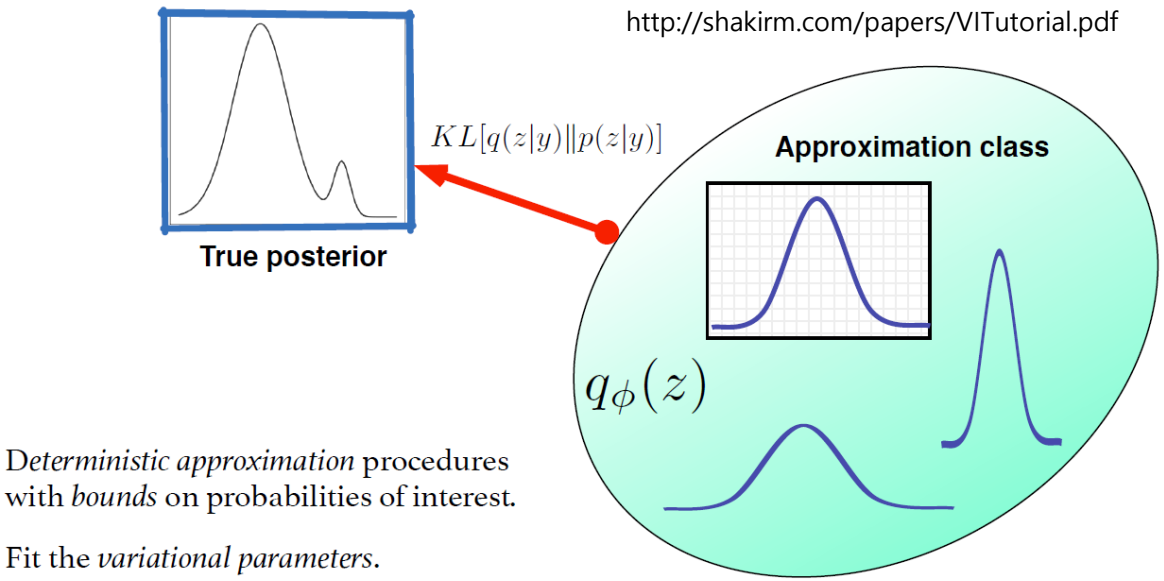
생성기에 대한 확률모델을 가우시안으로 할 경우, MSE관점에서 가까운 것이 더 p(x)에 기여하는 바가 크다.
MSE가 더 작은 이미지가 의미적으로도 더 가까운 경우가 아닌 이미지들이 많기 때문에 현실적으로 올바른 확률값을 구하기가 어렵다.

Tutorial on Variational Autoencoders : https://arxiv.org/pdf/1606.05908

VARIATIONAL INFERENCE | ELBO : Evidence LowerBOund

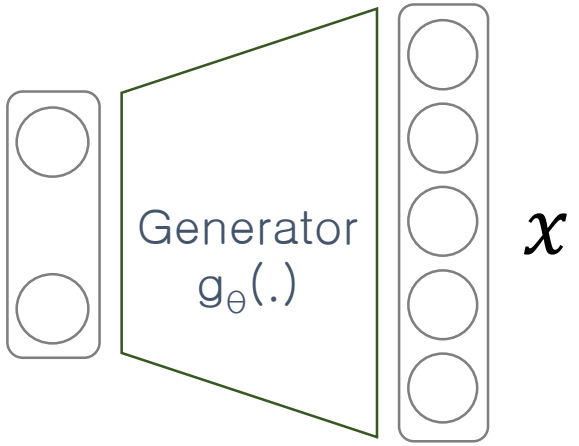
One possible solution : sampling z from $p(z|x)$

앞 슬라이드에서 Solution2가 가능하게 하는 방법 중 하나는 z를 정규분포에서 샘플링하는 것보다 x와 유의미하게 유사한 샘플이 나올 수 있는 확률분포 $p(z|x)$로 부터 샘플링하면 된다. 그러나 $p(z|x)$ 가 무엇인지 알지 못하므로, 우리가 알고 있는 확률분포 중 하나를 택해서 $(q_\phi(z|x))$ 그것의 파라미터값을 (λ) 조정하여 $p(z|x)$ 와 유사하게 만들어 본다. (Variational Inference)



[Variational Inference]

$$p(z|x) \approx q_\phi(z|x) \sim z$$

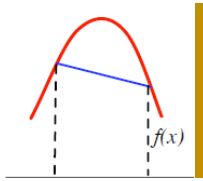


https://www.slideshare.net/haezoom/variational-autoencoder-understanding-variational-autoencoder-from-various-perspectives

VARIATIONAL INFERENCE | ELBO : Evidence LowerBOund

Relationship among $p(x), p(z|x), q_\phi(z|x)$: Derivation 1

$$\log\big(p(x)\big) = \log\left(\int p(x|z)p(z)dz\right) \geq \int \log\big(p(x|z)\big)p(z)dz \quad \leftarrow$$



[Jensen's Inequality]

For concave functions f(.)
f(E[x])≥E[f(x)]

f(.) = log(.) is concave

$$\log\big(p(x)\big) = \log\left(\int p(x|z)\frac{p(z)}{q_\phi(z|x)}q_\phi(z|x)dz\right) \geq \int \log\left(p(x|z)\frac{p(z)}{q_\phi(z|x)}\right)q_\phi(z|x)dz \leftarrow \text{Variational inference}$$

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

$$= \mathbb{E}_{q_\phi(z|x)}\big[\log\big(p(x|z)\big)\big] - KL\big(q_\phi(z|x)||p(z)\big)$$

$ELBO(\phi)$ Variational lower bound
Evidence lower bound (ELBO)

ELBO를 최대화하는 ϕ^*값을 찾으면 $\log\big(p(x)\big) = \mathbb{E}_{q_{\phi^*}(z|x)}\big[\log\big(p(x|z)\big)\big] - KL\big(q_{\phi^*}(z|x)||p(z)\big)$이다.

Relationship among $p(x), p(z|x), q_\phi(z|x)$: Derivation 2

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

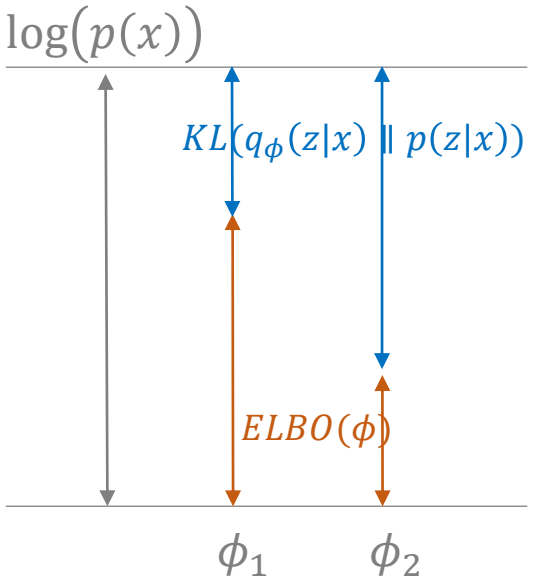
$$= \int \log\left(\frac{p(x,z)}{p(z|x)}\right) q_\phi(z|x)dz \quad \leftarrow p(x) = \frac{p(x,z)}{p(z|x)}$$

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

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

$$\underbrace{\qquad\qquad}_{ELBO(\phi)} \qquad \underbrace{\qquad\qquad\qquad\qquad}_{KL\left(q_\phi(z|x) \,\|\, p(z|x)\right)}$$

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.

Relationship among $p(x), p(z|x), q_\phi(z|x)$: Derivation 2

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

$$q_{\phi^*}(z|x) = \underset{\phi}{\operatorname{argmax}}\, ELBO(\phi)$$

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

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

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

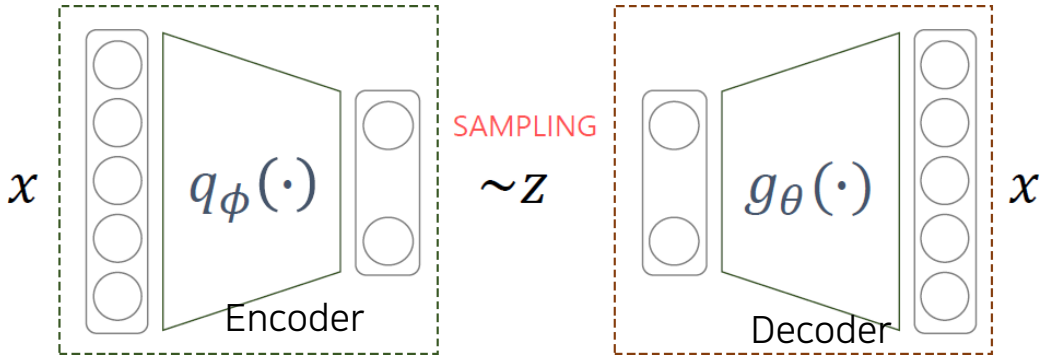
$$= \mathbb{E}_{q_\phi(z|x)}\big[\log\big(p(x|z)\big)\big] - KL\big(q_\phi(z|x)\big||p(z)\big)$$

앞 슬라이드에서의 KL과 인자가 다른 것에 유의

4. Variational Auto-Encoder(=V.A.E.)

세부적인 구조(1). Loss-Function

log(likelihood)를 Maximize하는 Loss-Function, 그리고 처음 보았던 Loss-Function과 동일한 모습



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

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

4. Variational Auto-Encoder(=V.A.E.)

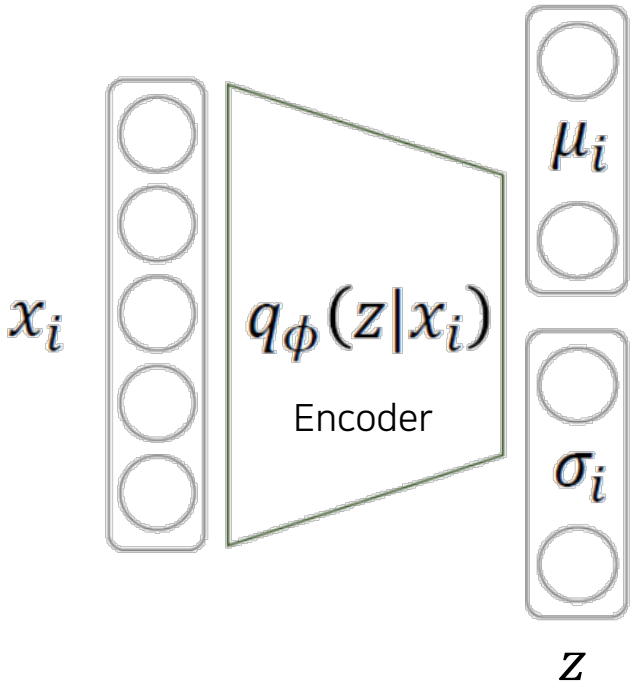
세부적인 구조(2). Optimization

Loss-Function에서 "Regularization"부분의 최적화(Optimization)

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

Regularization

Encoder 부분과 관련된 두가지 가정



Assumption 1

[Encoder : approximation class]

multivariate gaussian distribution with a diagonal covariance

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

Encoder쪽 r.v.인 $q_\phi(z|x_i)$의 Distribution은 항상 Normal-Distribution을 따른다고 가정한다. Multivariate-Gaussian은 벡터형태안에서 각 원소가 r.v.으로 이루어져 있음으로, 각 원소가 개별적인 r.v.으로 Normal-Distribution을 따른다는 것을 의미한다. Encoder는 항상 Normal-Dist임으로, Z(latent-vector)값으로 평균(μ_i)과 표준편차(σ_i)를 의미한다.

Assumption 2

[prior] multivariate normal distribution

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

본래 Encoder의 결과인 Z(latent-vector)에서 추정한 평균(μ_i)과 표준편차(σ_i)로 이루어진 N(μ_i, σ_i^2)에서 random-sampling을 해야하지만, back-propagation을 하기 위해 "Reparameterization Trick"을 사용해 표준정규분포인 N(0,1)에서 sample을 뽑아 σ_i를 곱하고, μ_i를 더해 그 값을 구현해준다.

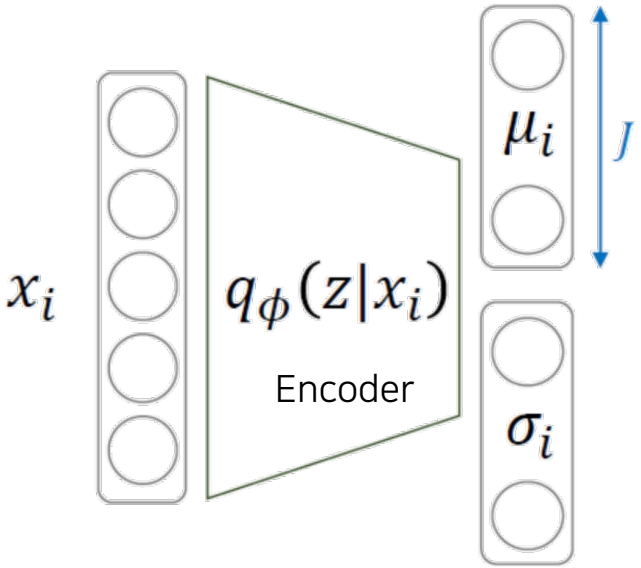
4. Variational Auto-Encoder(=V.A.E.)

세부적인 구조(2). Optimization

Loss-Function에서 "Regularization"부분의 최적화(Optimization)

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

Regularization



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

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

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

KLD for multivariate normal distributions

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

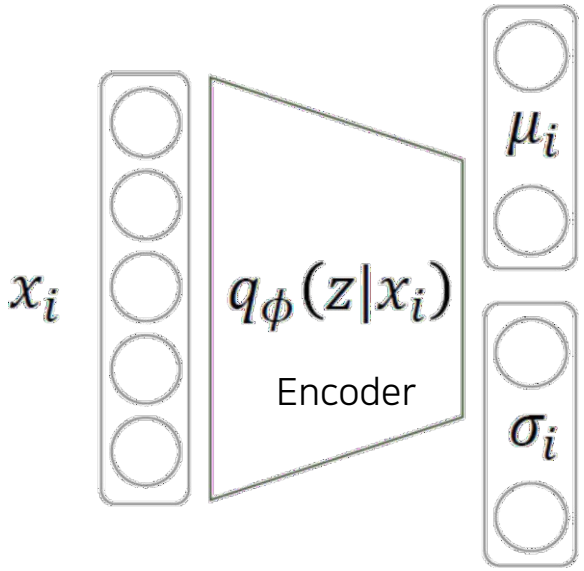
앞선 Assumption1,2에 의해 두개의 r.v.인 $q_\phi(z|x_i)$ & $p(z)$ 모두가 Normal-Dist임으로, KL-divergence를 위와 같이 직접적으로 구할 수가 있다.

4. Variational Auto-Encoder(=V.A.E.)

세부적인 구조(2). Optimization

Loss-Function에서 "Reconstruction Error"부분의 최적화(Optimization)

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

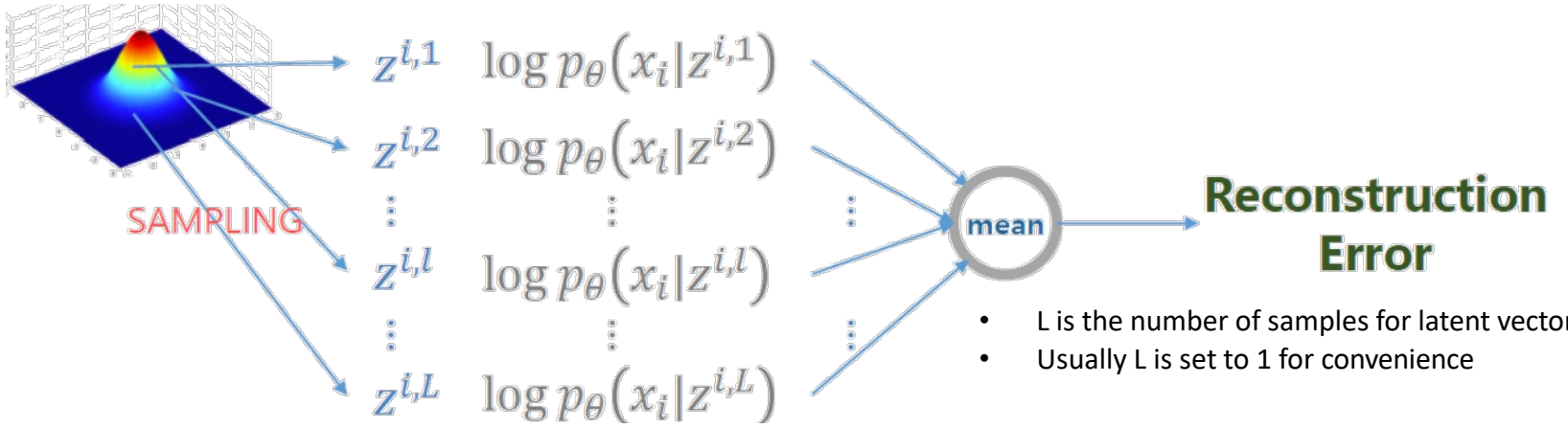


$$\mathbb{E}_{q_\phi(z|x_i)}\left[\log\left(p_\theta(x_i|z)\right)\right] = \int \log\left(p_\theta(x_i|z)\right) q_\phi(z|x_i)\,dz$$

Monte-carlo technique

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

적분을 하지 않고, L개를 뽑아서 평균을 낸다.



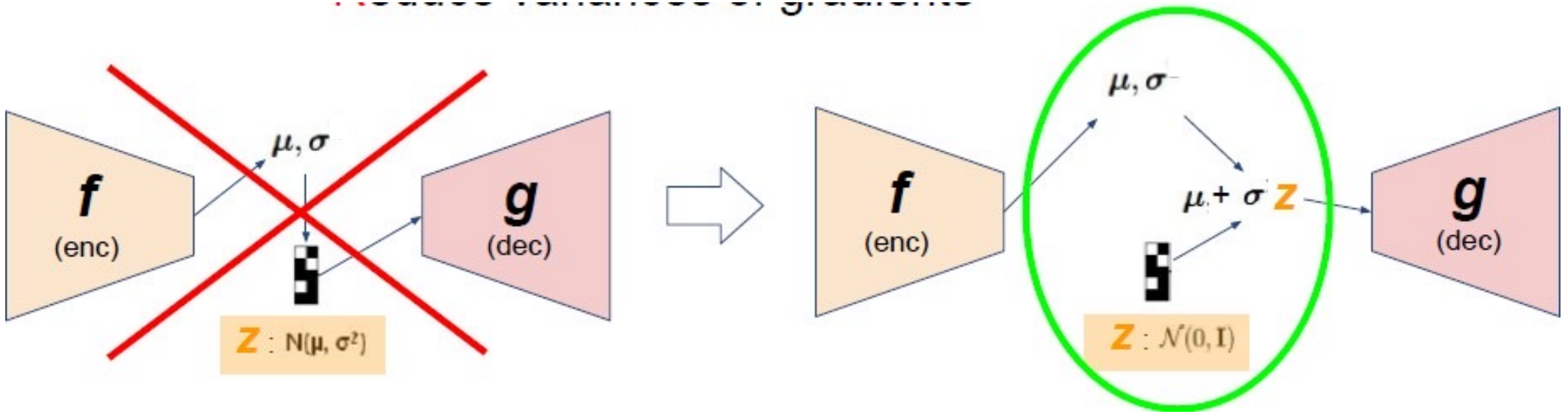
- L is the number of samples for latent vector
- Usually L is set to 1 for convenience

본래 Sampling을 하려면, 이전에 언급한 것처럼 Z(latent-vector)에서 추정한 평균(μ_i)과 표준편차(σ_i)로 이루어진 N(μ_i, σ_i^2)에서 random-sampling을 위처럼 해야한다.

4. Variational Auto-Encoder(=V.A.E.)

세부적인 구조(2). Optimization

Loss-Function에서 "Reconstruction Error"부분의 최적화(Optimization)



Sampling process

$$z^{i,l} \sim N(\mu_i, \sigma_i^2 I)$$

$$z^{i,l} = \mu_i + \sigma_i \odot \epsilon$$
$$\epsilon \sim N(0, I)$$

하지만 Gradient-Descent을 위한 back-propagation을 해야함으로, 왼쪽이 아닌 오른쪽과 같이 N(0,1)에서 sampling을 하는 "Reparameterization Trick"을 한다.

Same distribution!
But it makes backpropagation possible!

4. Variational Auto-Encoder(=V.A.E.)

세부적인 구조(2). Optimization

Loss-Function에서 "Reconstruction Error"부분의 최적화(Optimization)

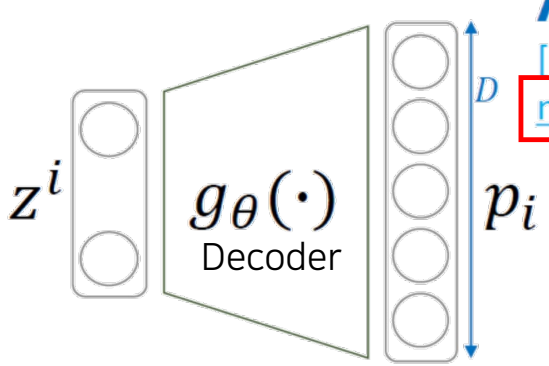
$$arg\min_{\theta,\phi} \sum_i -\mathbb{E}_{q_\phi(z|x_i)}\left[log\left(p(x_i|g_\theta(\ ))\right)\right] + KL(q_\phi(z|x_i)||p(z))$$

Reconstruction Error

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

Monte-carlo technique

L=1



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

Assumption 3-1

[Decoder, likelihood]

multivariate bernoulli or gaussain distribution

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

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

Decoder의 경우, $x_{i,j}|z^i$ 이라는 $r.v.$에 대하여 $Bernoulli\ Dist$ 혹은 $Normal\ Dist$로 가정할 수 있다. 위의 예시는 $Bernoulli\ Dist$로 가정했을 때, MLE로 나타낸 $Loss-Function$이 $Cross-Entropy$와 동일함을 보인 것이다.

4. Variational Auto-Encoder(=V.A.E.)

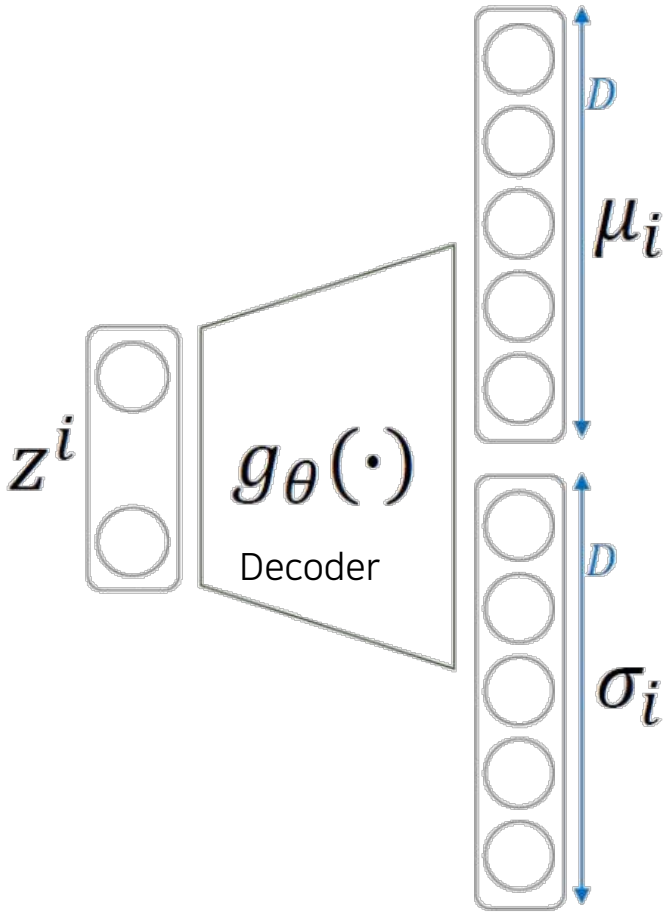
세부적인 구조(2). Optimization

Loss-Function에서 "Reconstruction Error"부분의 최적화(Optimization)

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

Reconstruction Error

$$\mathbb{E}_{q_\phi(z|x_i)}[log(p_\theta(x_i|z))] \approx \log\left(p_\theta\left(x_i|z^i\right)\right)$$



Assumption 3-2

[Decoder, likelihood]

multivariate bernoulli or gaussain distribution

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

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

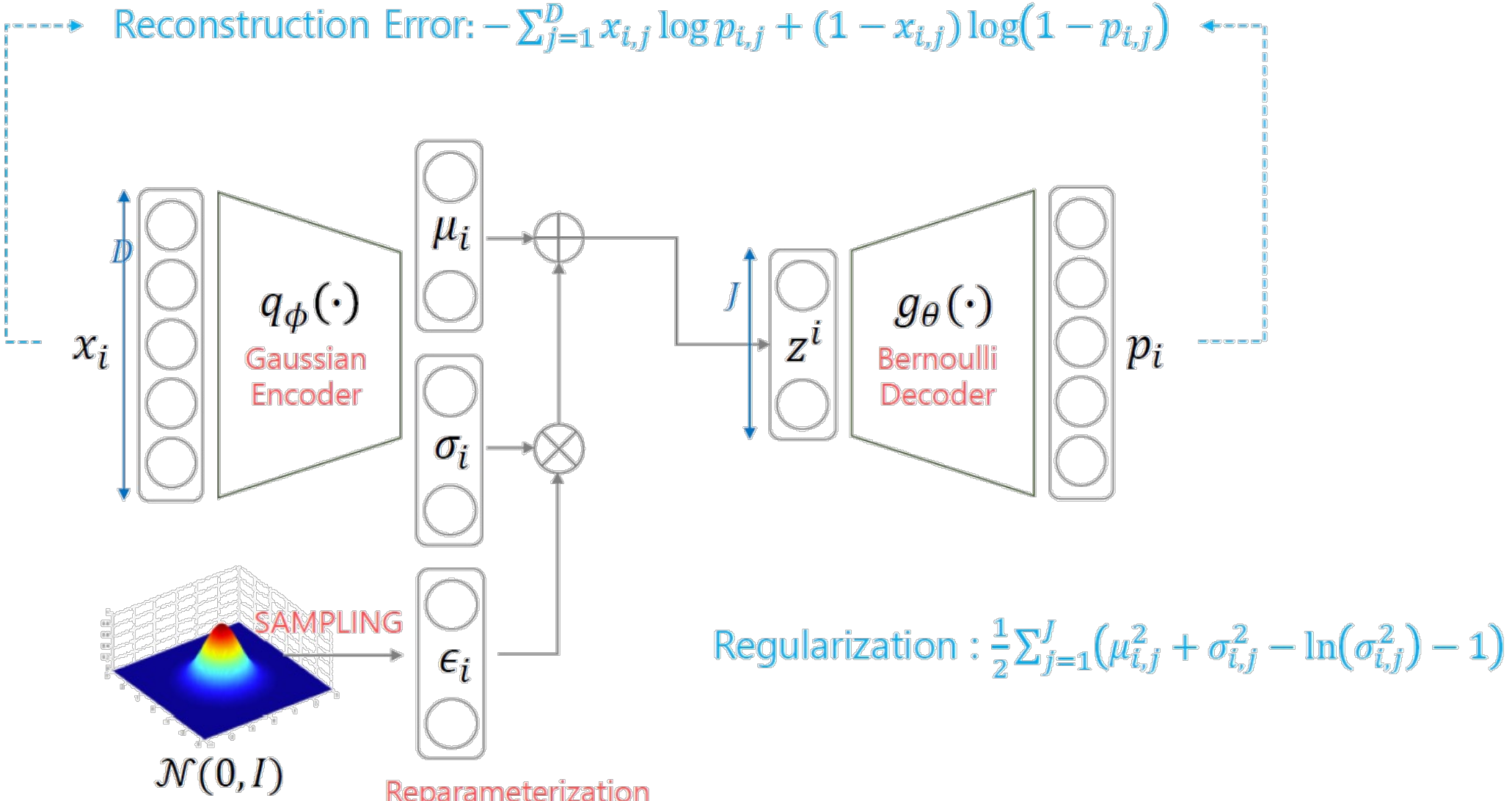
For gaussain distribution with identity covariance

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

Squared Error

4. Variational Auto-Encoder(=V.A.E.)

세부적인 구조(3). Gaussian Encoder + Bernoulli Decoder

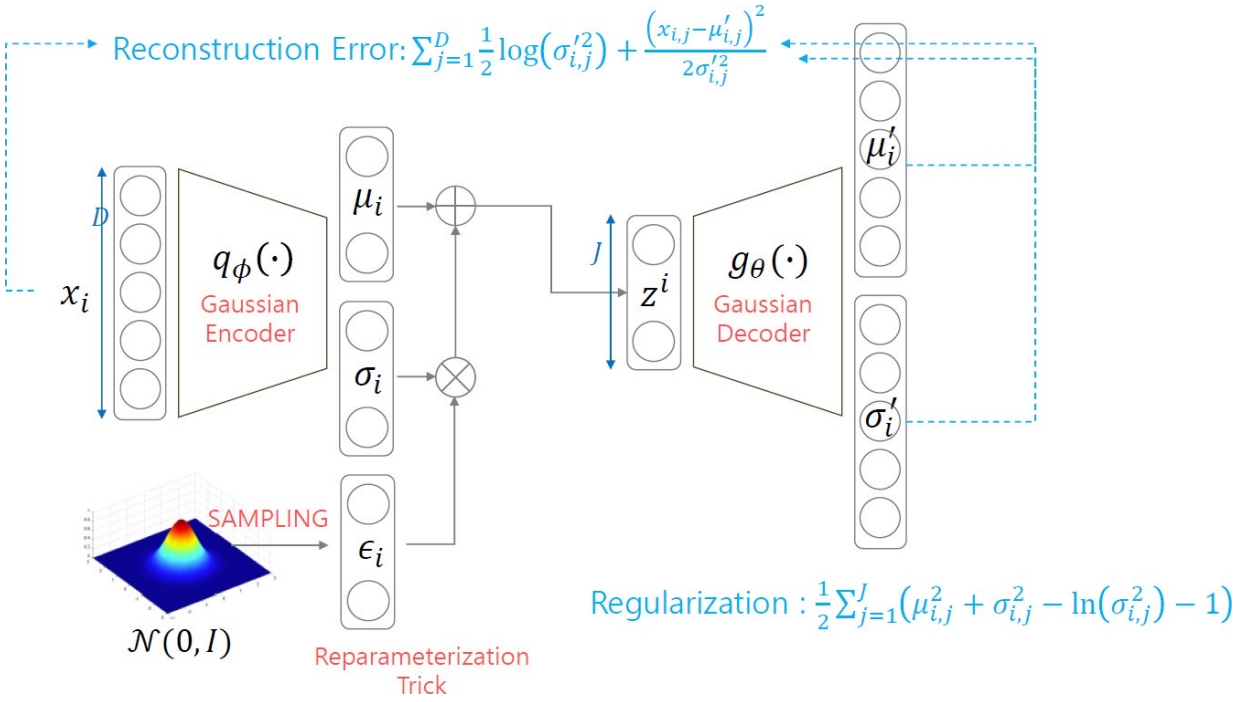


4. Variational Auto-Encoder(=V.A.E.)

세부적인 구조(3). Gaussian Encoder + Gaussian Decoder

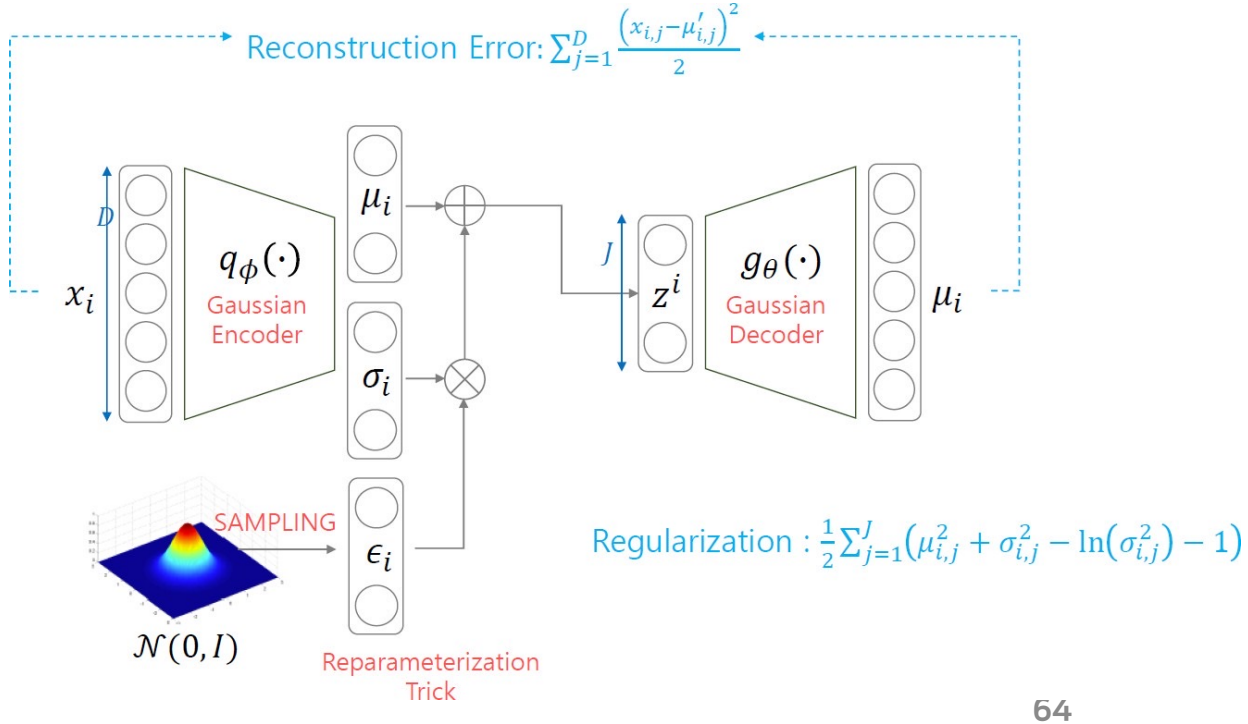
Gaussian Encoder + Gaussian Decoder

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



Gaussian Encoder + Gaussian Decoder with Identity Covariance

$$\sum_{j=1}^{D} (x_{i,j} - \mu_{i,j})^2$$ **Squared Error**

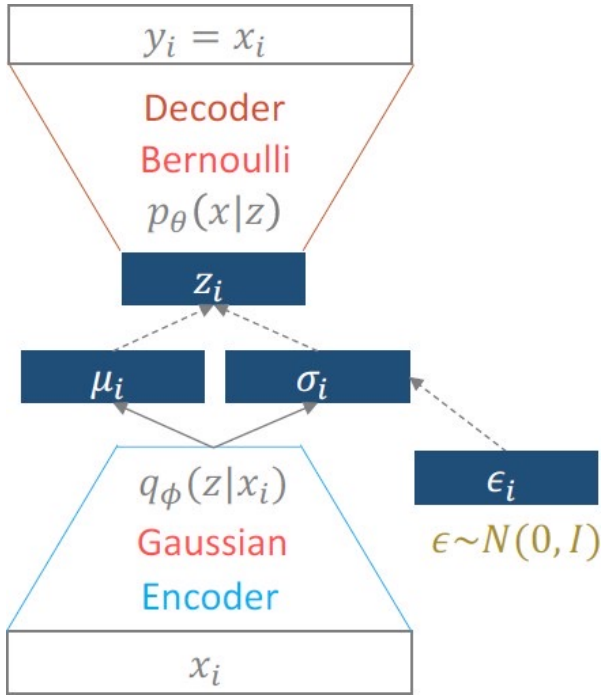


4. Variational Auto-Encoder(=V.A.E.)

세부적인 구조(4). V.A.E. Sumamry

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

입력과출력간의cross-entropy　　*Prior* 분포와의 다른 정도



- Probabilistic spin to traditional autoencoders ➔ allows generating data
- Defines an intractable density ➔ derive and optimize a (variational) lower bound

[VAE의 특징들]

1. Decoder가 최소한 학습 데이터는 생성해 낼 수 있게 된다.
➔ 생성된 데이터가 학습 데이터 좀 닮아 있다.

2. Encoder가 최소한 학습 데이터는 잘 latent vector로 표현할 수 있게 된다.
➔ 데이터의 추상화를 위해 많이 사용된다.

4. Variational Auto-Encoder(=V.A.E.)

세부적인 구조(4). V.A.E. Sumamry

$$L_i(\phi,\theta,x_i) = \underbrace{-\mathbb{E}_{q_\phi(z|x_i)}[\log\left(p\left(x_i|g_\theta(z)\right)\right)]}_{\text{Reconstruction Error}} \underbrace{+KL(q_\phi(z|x_i)||p(z))}_{\text{Regularization}} \Rightarrow argmax\ \text{ELBO}(\phi)$$

Reconstruction Error
원데이터에 대한 Log Likelihood

Regularization
다루기 쉬운 확률 분포 중 선택해서 변이추론을 위한 근사 class중 선택하여 유사해야 한다는 조건을 부여함.

코딩에 적용된 수식

[*Regularization* : $Kullback-leibler\ divergence$]

$$KL(q_\phi(z|x_i)||p(z)) = \frac{1}{2}\sum_{j=1}^{J}\left(\mu_{i,j}^2+\sigma_{i,j}^2-\ln(\sigma_{i,j}^2)-1\right)$$

[*Reconstruction Error*]

$$-\mathbb{E}_{q_\phi(z|x_i)}\left[\log\left(p\left(x_i|g_\theta(z)\right)\right)\right] = \int \log\left(p\left(x_i|g_\theta(z)\right)\right) \approx \underbrace{\frac{1}{L}\sum_{z^{i,l}} log(p_\theta(x_i|z^{i,l}))}_{\text{Monte-carlo technique}} \approx \log(p_\theta(x_i|z^{i,l})) = \sum_{j=1}^{D} x_{i,j}logp_{i,j} + (1-x_{i,j})log(1-p_{i,j})$$

For Bernoulli = cross-entropy

For Gaussian distibition
= mean square Error

DATA
SCIENCE LAB

05. 여러가지 V.A.E.와 그 활용

5. 여러가지 V.A.E.와 그 활용

To be Continued… 기회가 된다면

Reference

- https://youtu.be/o_peo6U7IRM
- http://www.smartdesignlab.org/dl_aischool_2021.html
- Contributors: 김성신, 유소영, 이성희, 김은지
- Andrew Ng의ML Class (www.holehouse.org/mlclass/)
- Fei-Fei Li & Justin Johnson & Serena Yeung, CS231n: Convolutional Neural Networks for Visual Recognition, Stanford (http://cs231n.stanford.edu/)
- Stefano Ermon & Aditya Grover, CS 236: Deep Generative Models , Stanford (https://deepgenerativemodels.github.io/)
- 모두를위한딥러닝(https://hunkim.github.io/ml/)
- 모두를위한딥러닝시즌2 (https://deeplearningzerotoall.github.io/season2/lec_tensorflow.html)
- 이활석, Autoencoders (https://www.slideshare.net/NaverEngineering/ss-96581209)
- 최윤제, 1시간만에GAN(Generative Adversarial Network) 완전정복하기(https://www.slideshare.net/NaverEngineering/1-gangenerative-adversarial-_network?qid=c53ce33f-6643-4437-8e93-88776c9cebb1&v=&b=&from_search=5)
- 김성범, [핵심머신러닝] Principal Component Analysis (PCA, 주성분분석) (https://youtu.be/FhQm2Tc8Kic)

DATA SCIENCE LAB

발표자 김남훈 010-7163-2566
E-mail: ksouth0413@naver.com