실시간 스타일 변환 어플리케이션 개발과정

생성모델I

Introduction

GAN을 사용하면 컴퓨터가 새로운 데이터를 생성할 수 있기 때문에 딥러닝 분야의 가장 중요한 혁신으로 간주되고 있다.

우리는 이 과정에서 다음의 순서대로 내용을 다룹니다.

- 새로운 데이터를 합성하기 위한 생성 모델 소개
- 오토 인코더(Auto Encoder), 변이형 오토 인코더(Variational Auto Encoder), 그리고 GAN
- GAN의 구성 요소 이해하기
- 손글씨 숫자를 생성하는 간단한 GAN 모델 구현하기
- 전치 합성곱(Transfposed convolution)과 배치 정규화(Batch Normalization, BN)의 이해
- GAN 성능 향상 : 심층 합성곱 GAN과 바서슈타인 거리(Wasserstein distance)를 사용한 GAN

생성 모델의 기초

GAN의 주요한 목적은 훈련 데이터셋과 동일한 분포를 가지는 새로운 데이터를 합성하는 것이다. 따라서 GAN의 원본 형태는 레이블 데이터가 필요하지 않으므로 머신 러닝 작업 중 비지도 학습으로 정의된다. 하지만, 원본 GAN을 확장한 것은 비지도 학습과 지도 학습 양쪽에서 모두 볼 수 있다.

일반적인 GAN의 개념은 이안 굿펠로우와 동료들이 심층 신경망을 사용하여 새로운 이미지를 합성하는 방법으로 2014년 처음 발표했다. 이 논문에서 제안한 초기 GAN구조는 다층 퍼셉트론과 비슷한 완전 연결 층을 기반으로 하여 낮은 해상도의 MNIST 손글씨 숫자를 생성하도록 훈련했다. 이것 만으로도 이 기술의 가능성을 증명하기에는 충분했다.

처음 소개된 이후 원저자는 물론 많은 연구자가 헤아릴 수 없을 정도로 많은 개선 버전을 제시했고 공학과 과학의 여러 분야에서 다양한 어플리케이션을 만들었다.

생성 모델의 응용분야

예를 들어 컴퓨터 비전에서는 이미지 변환(image-to-image translation), 이미지 초해상도(Image super-resolution: 낮은 해상도의 이미지를 높은 해상도의 이미지로 변환), 이미지 인페인팅(image inpainting:이미지에서 누락된 부분을 재구성하는 방법을 학습)등 많은 어플리케이션에서 GAN을 사용한다. 최근 GAN 연구 성과는 새로운 고해상도 얼굴 이미지를 생성할 수 있는 모델을 만들었다.

GAN이 생성한 고해상도 얼굴 이미지를 https://www.thispersondoesnotexist.com/에서 확인할 수 있다.

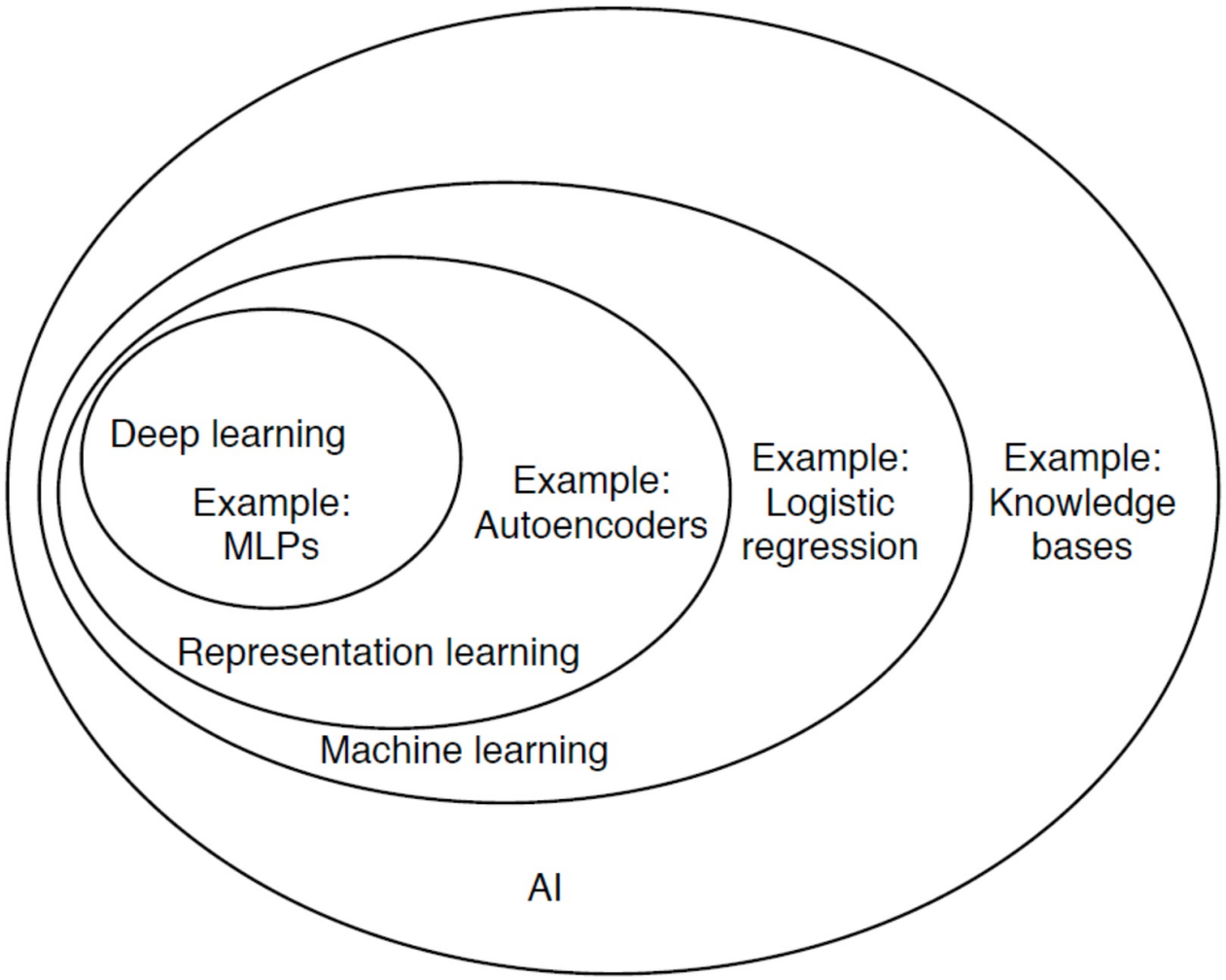
오토 인코더

GAN의 동작 방식을 설명하기 전에 먼저 훈련 데이터를 압축하고 해제할 수 있는 오토인코더를 다루어 봅시다. 기본 오토 인코더는 새로운 데이터를 생성할 수 없지만, 오토인코더를 이해하면 다음 절에서 배울 GAN을 이해하는데 도움이 됩니다.

오토 인코더는 인코더 신경망과 디코더 신경망 두 개가 연결되어 구성됩니다. 인코더 신경망은 샘플 x에 연관된 d차원의 입력 특성 벡터를 받아 p차원의 벡터로 인코딩합니다.

정리하면, 인코더는 $z = f(x)$ 함수를 모델링하는 방법을 배우는 역할을 합니다. 인코딩된 벡터 z를 잠재 벡터(latent vector) 또는 잠재 특성 표현이라고 합니다. 일반적으로 잠재 벡터의 차원은 입력 샘플의 차원보다 작습니다. 즉, p<d입니다. 따라서 **인코더가 데이터 압축 기능**을 한다고 말합니다.

오토 인코더



http://videolectures.net/kdd2014_bengio_deep_learning/

[KEYWORDS]

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

생성 모델의 분류 체계

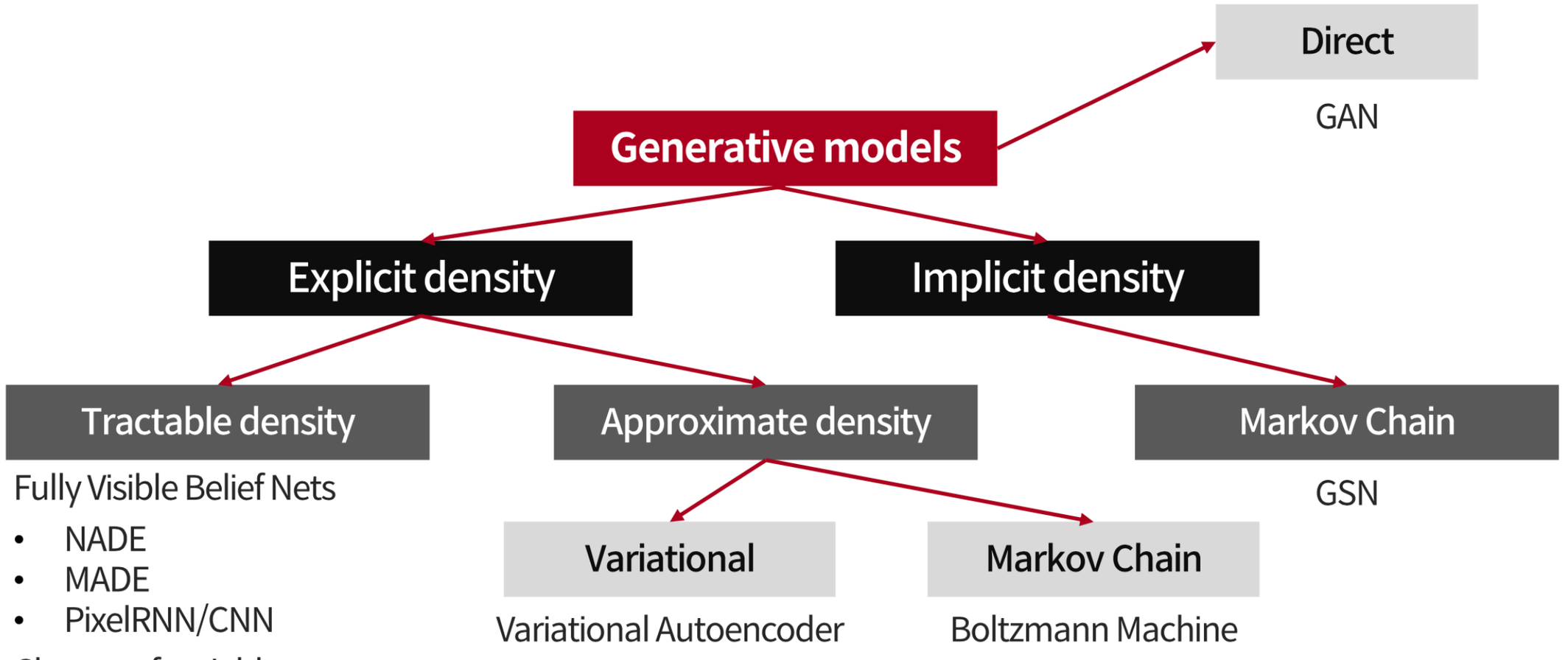
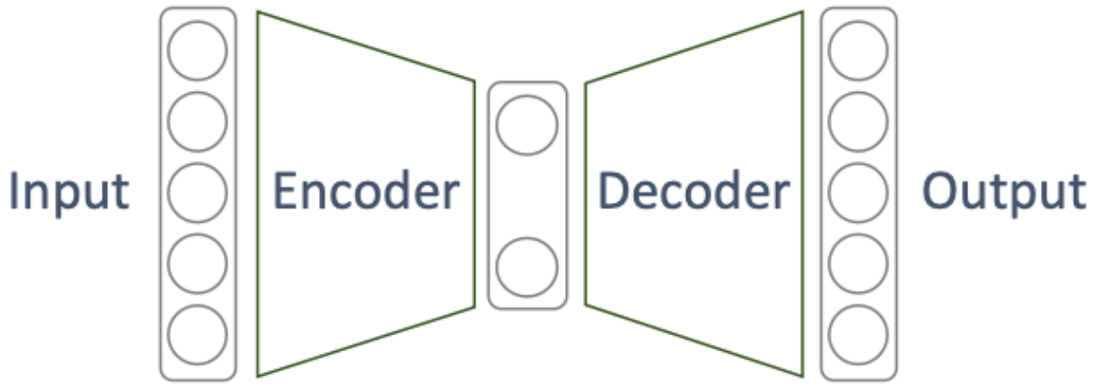


Figure from Ian Goodfellow,s Tutorial on Generative Adversarial Networks, 2017.

오토 인코더 – 4개의 키워드



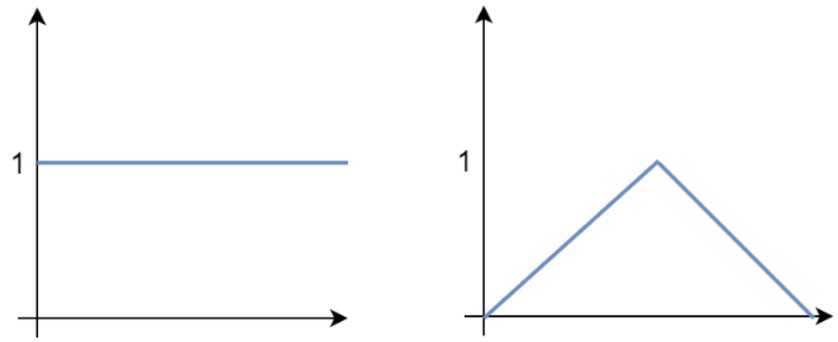
[4 MAIN KEYWORDS]
1. Unsupervised learning
2. Manifold learning
3. Generative model learning
4. ML density estimation

오토 인코더를 학습할 때:
학습 방법은 비지도 학습 방법을 따르며, ⇢ Unsupervised learning
Loss는 Negative Maximum-likeihood로 해석된다. ⇢ ML density estimation

학습된 오토 인코더에서:
인코더는 차원 축소 역할을 수행하며, ⇢ Manifold learning
디코더는 생성 모델의 역할을 수행한다. ⇢ Generative model learning

오토 인코더 – Likeihood

Likelihood란, 데이터가 특정 분포로부터 만들어졌을(generate) 확률을 말한다. 예를 들어, X = (1, 1, 1, 1, 1)이라는 데이터가 있다고 하자. 그리고 아래와 같이 두 분포가 있다고 하자.



두 개의 분포

우리의 데이터 X는 어떤 분포를 따를 확률이 더 높을까? 당연히 왼쪽 분포를 따를 확률이 더 높을 것이다. 이런 상황에서 우리는 왼쪽 분포의 데이터 X에 대한 likelihood가 더 높다고 표현할 수 있다. 그러므로 likelihood는 다음과 같은 식으로 표현할 수 있다.

$$L(\theta) = p(X|\theta)$$

likelihood의 표현식. θ의 parameter를 가지는 분포.

조건부 확률

오토 인코더 – 결합 확률(joint probability)

사건 A와 사건 B가 동시에 발생할 확률, 즉 A도 진실이고, B도 진실이므로 사건 A와 B의 교집합의 확률을 계산하는 것과 같다.

$$\mathrm{P}(A\ and\ B) = P(A \cap B)$$

만약 두 사건이 서로 독립적이라면 아래와 같은 수식 표현이 가능하다.

$$P(A \cap B) = P(A) \cdot P(b)$$

주사위 한 개를 두 번 던졌을 때,

3(사건 A), 6(사건 B)이 순서대로 나올 확률 같은 경우이다.

$$P(A \cap B) = P(A) \cdot P(b) = \frac{1}{6} \times \frac{1}{6} = \frac{1}{36}$$

오토 인코더 – 조건부확률(conditional probability)

사건 B가 일어난 상태에서 사건 A가 일어날 확률.

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

만약 두 사건이 서로 독립적이라면, 아래와 같이 정리 된다.

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A)P(B)}{P(B)} = P(A)$$

참고사이트 : 결합확률과 조건부확률

오토 인코더 – Likeihood

그렇다면 이 likelihood는 어떻게 계산할 수 있을까? 우리의 distribution(분포)이 θ=(μ, σ) 의 parameter를 가지고 있는 정규분포라고 가정하자. 그러면 한 개의 데이터가 이 정규분포를 따를 확률은 다음과 같이 계산할 수 있을 것이다. (그냥 정규 분포의 PDF에 x_n 값을 넣은 것이다.)

$$p(x_n \mid \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\{-\frac{(x_n - \mu)^2}{2\sigma^2}\}$$

데이터 x_n이 θ=(μ, σ)의 parameter를 가지는 정규분포를 따를 확률

그렇다면 모든 데이터들이 독립적(independent)이라고 가정하면 다음과 같은 likelihood 식을 얻을 수 있다.

$$L(\theta) = p(X|\theta) = \prod_{n=1}^{N} p(x_n|\theta)$$

likelihood 계산식

참고 사이트 : https://jinseob2kim.github.io/probability_likelihood.html

오토 인코더 – Maximum Likelihood

Back to Machine Learning Problem

01. Collect training data

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

02. Define functions

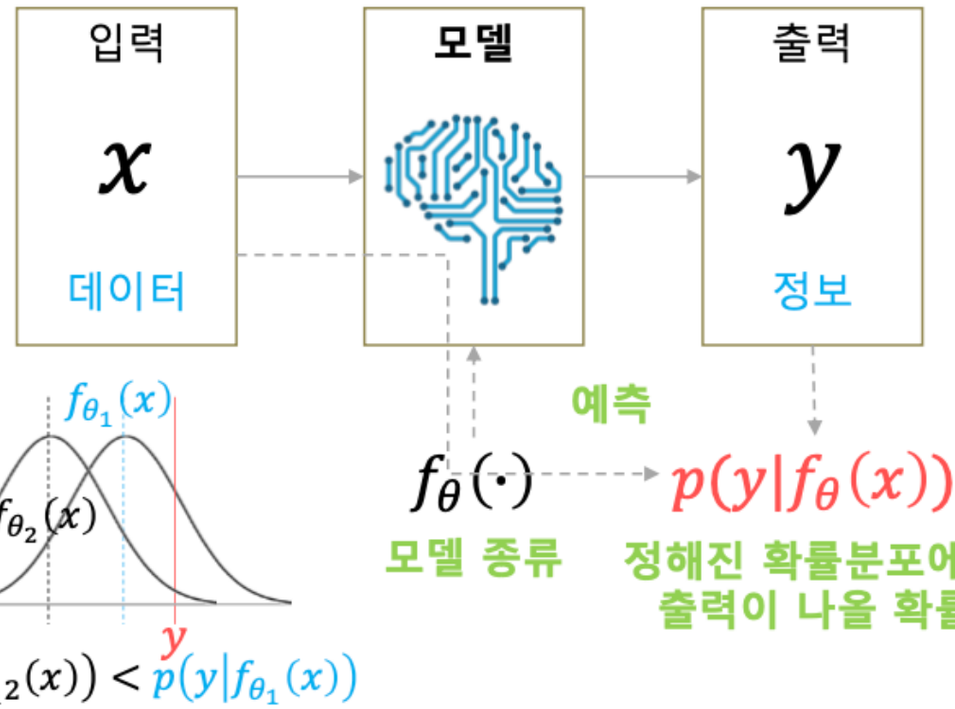
- Output : $f_\theta(x)$
- Loss : $-\log(p(y|f_\theta(x)))$

03. Learning/Training

Find the optimal parameter

04. Predicting/Testing

Compute optimal function output



$$\theta^* = \underset{\theta}{\operatorname{argmin}}[-\log(p(y|f_\theta(x)))]$$ 주어진 데이터를 제일 잘 설명하는 모델 찾기

$$y_{new} \sim p(y|f_{\theta^*}(x_{new}))$$ 고정 입력, 고정/다른 출력

참고 : Autoencoders - A way for Unsupervised Learning of Nonlinear Manifold - 이활석

오토 인코더 실습

Colab에서 Auto Encoder 실습

Variational Auto Encoder(VAE) 사전지식

VAE는 Auto Encoder와는 다르게 Generative Model이다.
Generative Model이란 Training데이터가 주어졌을 때, 이 데이터가 가지는 데이터 분포와 같은 분포를 갖는 새로운 데이터를 생성하는 모델을 말한다.

VAE를 이해하기 위해서는 몇가지 사전 지식을 필요로 한다.

베이지안 확률(Bayesian Probability)

일어나지 않은 일에 대한 확률을 사건과 관련이 있는 여러 확률들을 이용해 우리가 알고 싶은 사건을 추정하는 것이 베이지안 확률이다.

Variational Auto Encoder(VAE) – 확률

빈도 확률(Frequentist Probability) VS 베이지안 확률(Bayesian Probability)

빈도확률

동전의 앞면이 나올 확률은 일단 동전을 던지면 된다. 10번 던졌을 때 4개가 앞면이면, 앞면이 나올 확률 2/5, 100번을 던졌을 때 45개가 앞면이면 앞면이 나올 확률 9/20이다. 이런 방법으로 시행횟수를 반복하여 빈도수를 측정하면 된다.

베이지안 확률

화산이 폭발할 확률을 빈도 확률로 계산하기는 어렵다. 우리가 동전을 던지듯이 화산을 폭발 시킬수 없기에 빈도 확률 방법으로는 신뢰할 만한 값을 얻기 어렵다. 세상에는 반복할 수 없는 사건이 무수히 많고, 빈도 확률의 개념을 그러한 사건에 적용할 수 없다. 일어나지 않은 일에 대한 확률을 불확실성의 개념. 즉, 사건과 관련 있는 여러 확률을 이용해 새롭게 일어날 사건을 추정하는 것이 베이지안 확률이다.

베이지안 확률

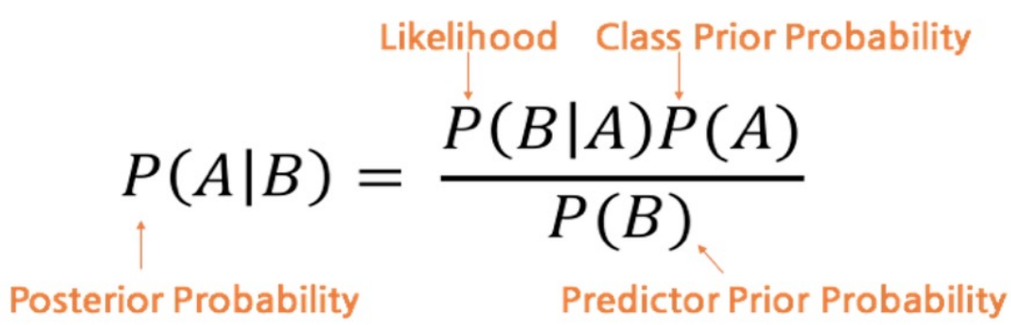
베이지안 확률의 정의

베이즈 정리라 불리며, 종속적(의존적)관계에 놓인 사건들을 기반으로 확률을 구함.

두 확률 변수의 사전 확률과 사후 확률 사이의 관계를 나타내는 정리

사전 확률 P(A)와 우도 확률 P(B|A)를 안다면 사후 확률 P(A|B)를 알 수 있다.

베이지안 확률은 아래 조건부 확률로 나타내며, 정보를 업데이트 하면서 사후 확률 P(A|B)를 구하는 것이다.



P(A), 사전 확률 : 결과가 나타나기 전에 결정되어 있는 A(원인)의 확률

P(B|A), 우도 확률(Likelihood Probability) : A(원인)가 발생했다는 조건하에 B가 발생할 확률

P(A|B), 사후 확률(Posterior Probability) : B(결과)가 발생했다는 조건하에 A(원인)가 발생할 확률

베이지안 확률 계산식

앞의 정의가 나오는 계산식의 과정은 다음과 같다.

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

$$P(A|B)\,P(B) = P(A \cap B) = P(B|A)\,P(A)$$

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

$$p(A|B) = \frac{p(B|A)\,p(A)}{p(B|A)\,p(A) + p\,(B|A')\,p(A')}$$

$$P(B) = P(B \cap A) + P(B \cap A')$$

확률 분포(Probability distribution)의 추정

주사위를 던졌을 때 이상적 확률

확률 변수(Random variable)

확률 질량 함수(Probability mass function)

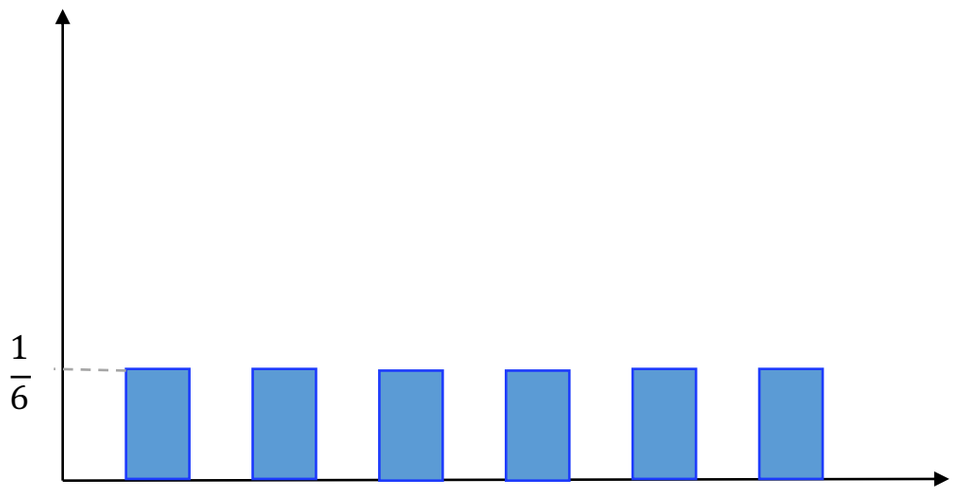
X	1	2	3	4	5	6
P(X)	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$

Probability Mass VS Density Functions

1. Probability Mass Function — Discrete

2. Probability Density Function — Continuous

두 함수는 연속성을 갖는지, 이산성을 갖는지에 따라 Mass Function과 Density Function으로 나뉘어진다.

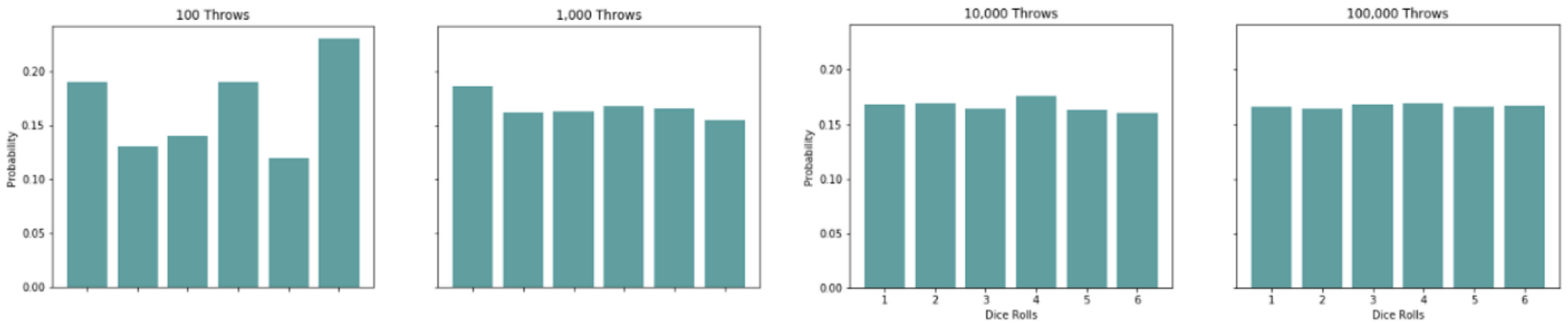


참고 : https://towardsdatascience.com/probability-mass-and-density-functions-eab86a81d021

확률 분포(Probability distribution)의 추정

주사위를 던졌을 때 실험 확률

주사위를 던진 횟수가 증가할수록 이상치(ideal)에 수렴한다.



주사위를 던진 횟수

기댓값(Expected Value)

$$E(X) = \mu = \sum_i p(x_i)x_i$$

이상적인 주사위의 기댓값

$$E(X) = 1 * \frac{1}{6} + 2 * \frac{1}{6} + 3 * \frac{1}{6} + 4 * \frac{1}{6} + 5 * \frac{1}{6} + 6 * \frac{1}{6} = 3.5$$

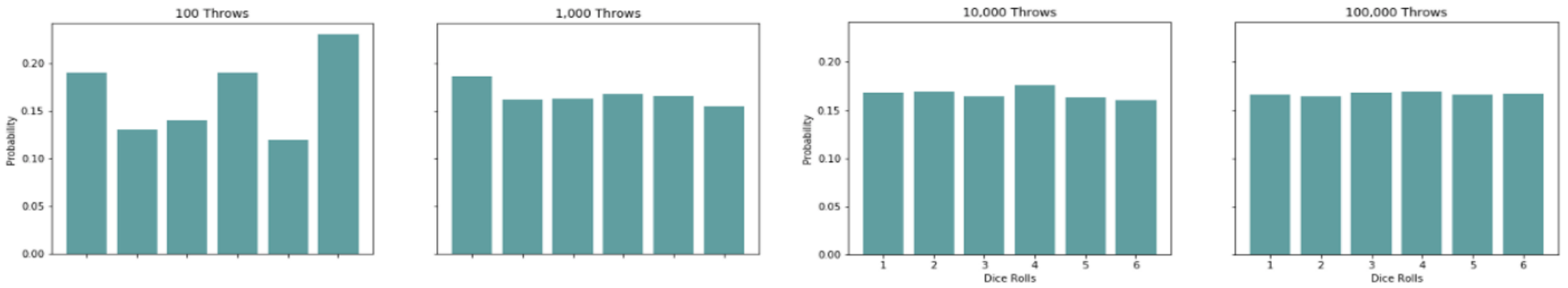
주사위 100번 던진 통계치의 기댓값

$$E(X) = 1 * 0.19 + 2 * 0.13 + 3 * 0.14 + 4 * 0.19 + 5 * 0.12 + 6 * 0.23 = 3.61$$

참고 : https://towardsdatascience.com/probability-mass-and-density-functions-eab86a81d021

확률 분포(Probability distribution)의 추정

Variance



$$E((X-\mu)^2) = \sigma^2 = \sum_i p(x_i)(x_i-\mu)^2$$

$$Variance = 0.19 * (1-3.61)^2 + 0.13 * (2-3.61)^2 + 0.14 * (3-3.61)^2$$
$$+ 0.19 * (4-3.61)^2 + 0.12 * (5-3.61)^2 + + 0.23 * (6-3.61)^2 = 3.26$$

참고 : https://towardsdatascience.com/probability-mass-and-density-functions-eab86a81d021

확률 분포(Probability distribution)의 추정

Probability Density Function — Continuous

확률 밀도 함수(PDF)

$$P(a \leq x \leq b) = \int_a^b f(x)dx \geq 0$$

기댓값(Expected Value)

$$E(X) = \mu = \int_{-\infty}^{+\infty} p(x)xdx$$

분산(Variance)

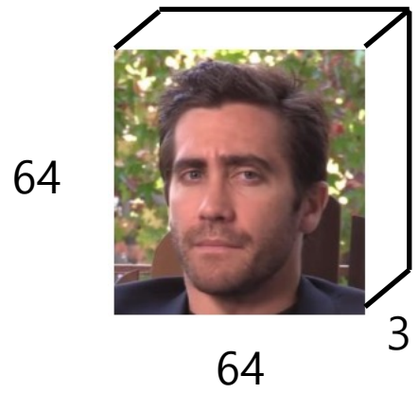
$$E((X-\mu)^2) = \sigma^2 = \int_{-\infty}^{+\infty} p(x)(x-\mu)^2dx$$

참고 : https://towardsdatascience.com/probability-mass-and-density-functions-eab86a81d021

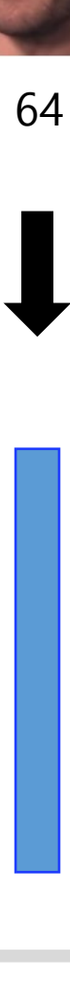
확률 분포(Probability distribution)의 추정

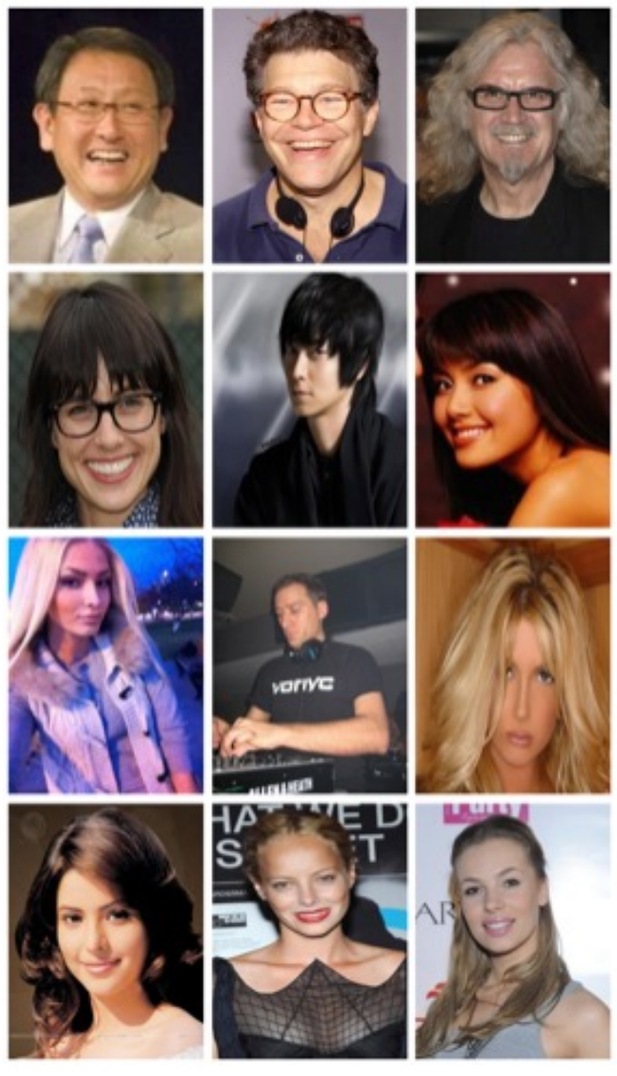
앞에서 다룬 주사위의 확률 변수x가 스칼라 값이었다면,

이번에는 이미지를 가정해 봅시다.



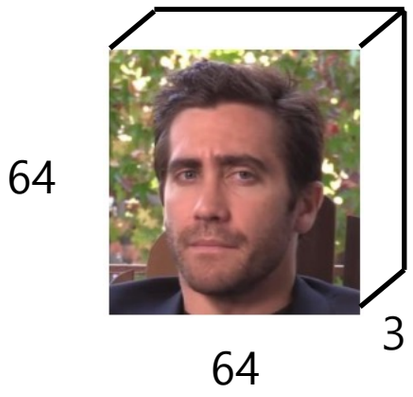
64x64x3



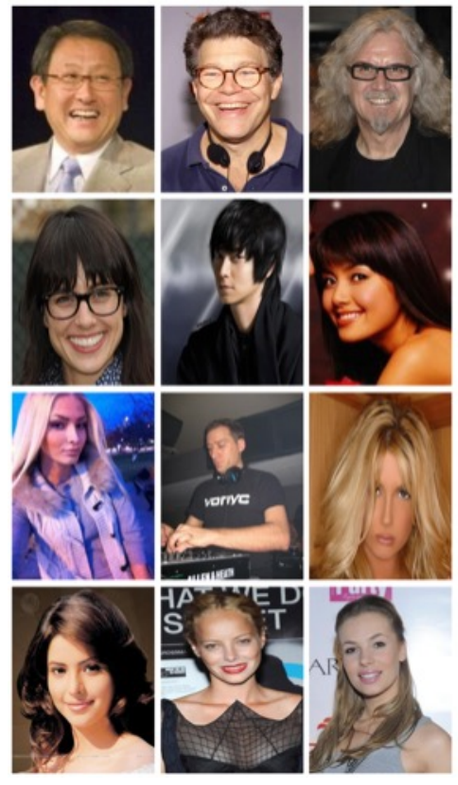


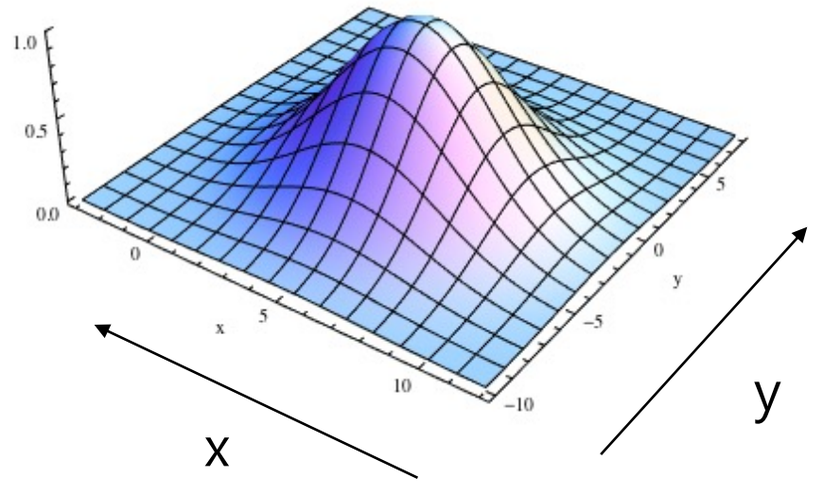
확률 분포(Probability distribution)의 추정

앞에서 다룬 주사위의 확률 변수x가 스칼라 값이었다면,
이번에는 이미지를 가정해 봅시다.



이미지가 64x64x3의 차원을 갖는다면

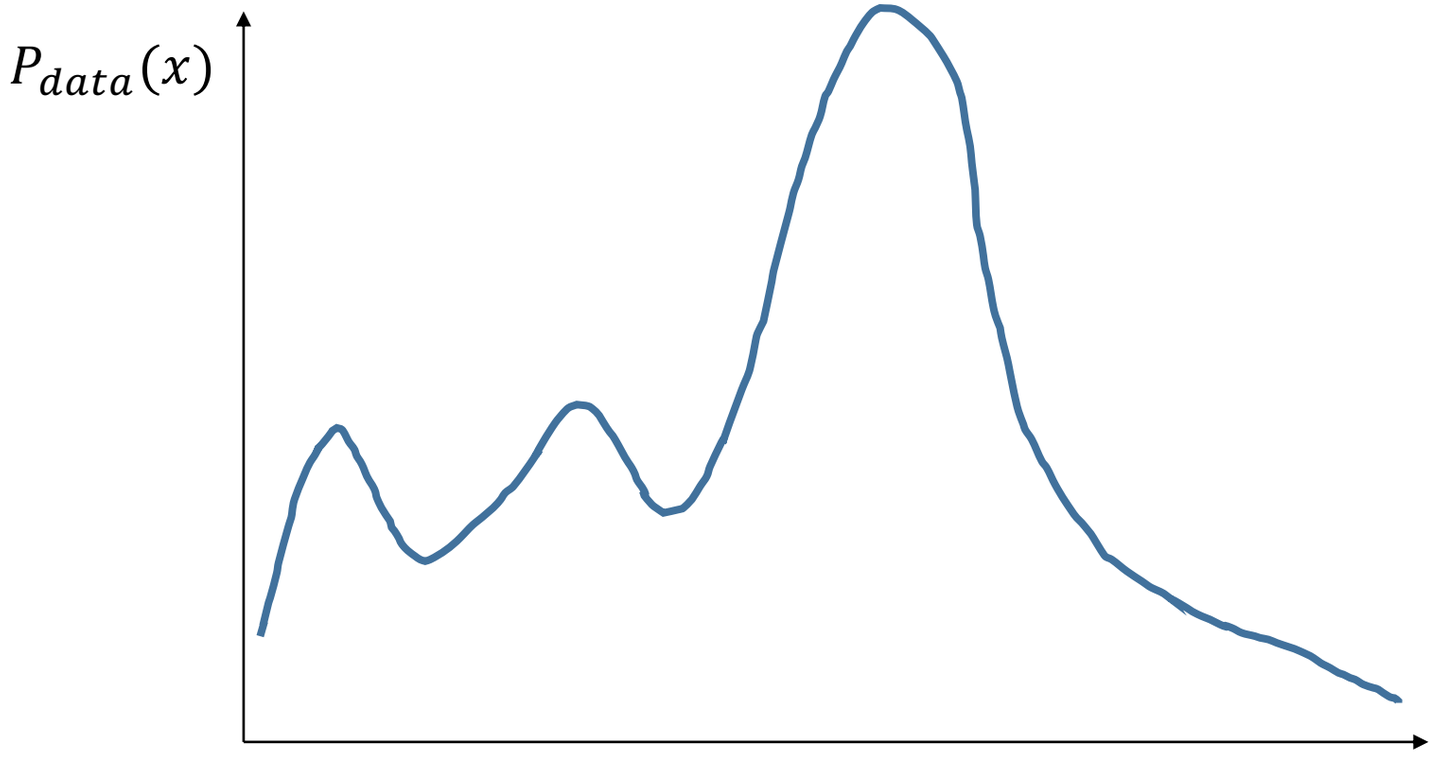




64x64x3

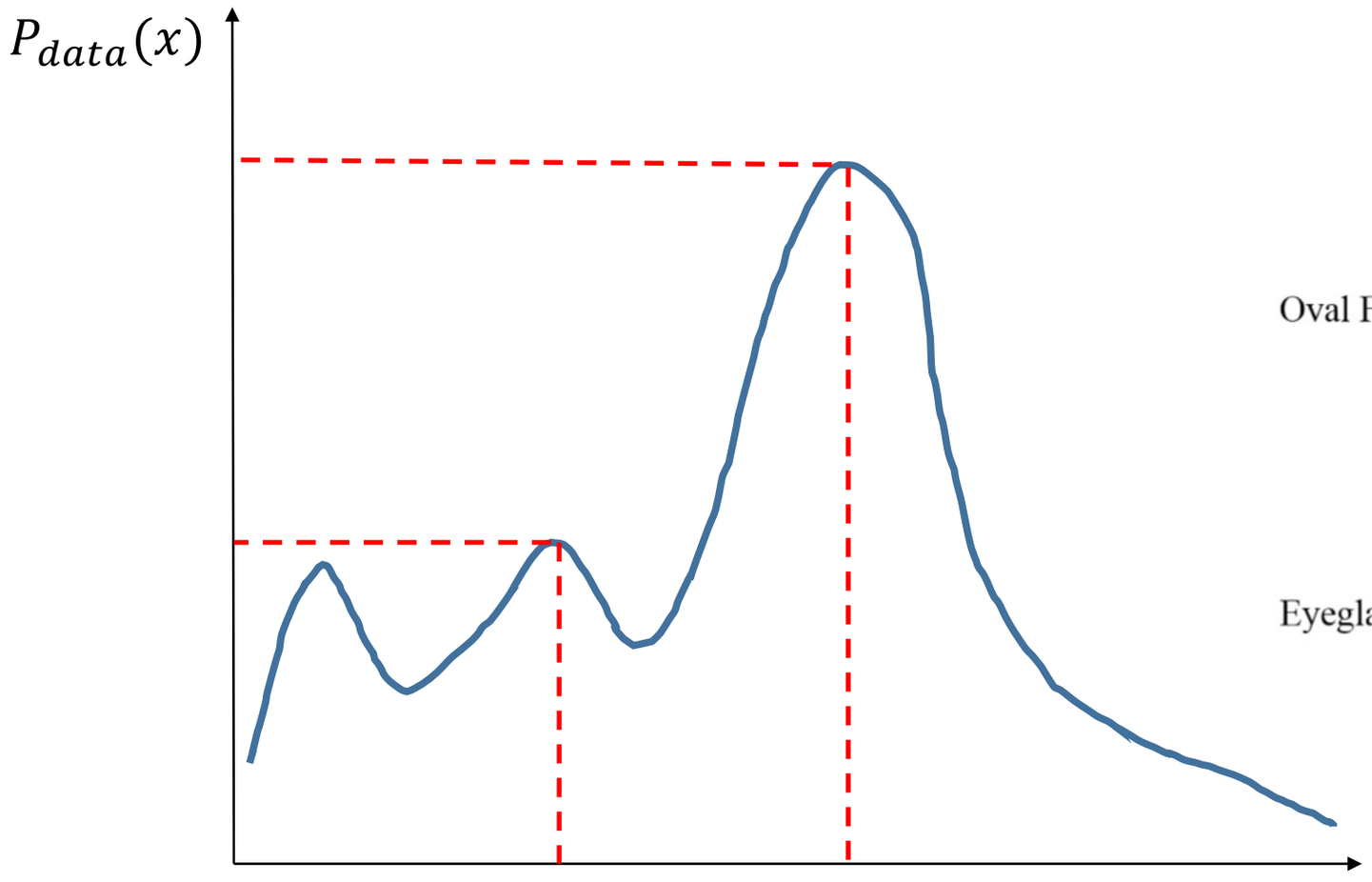
이미지들에 대한 확률 분포를 미리 알고 있다면 우리는 쉽게 추정할 수 있다. 위의 확률 분포 함수를 reshape해서 1차원으로 변경해주면

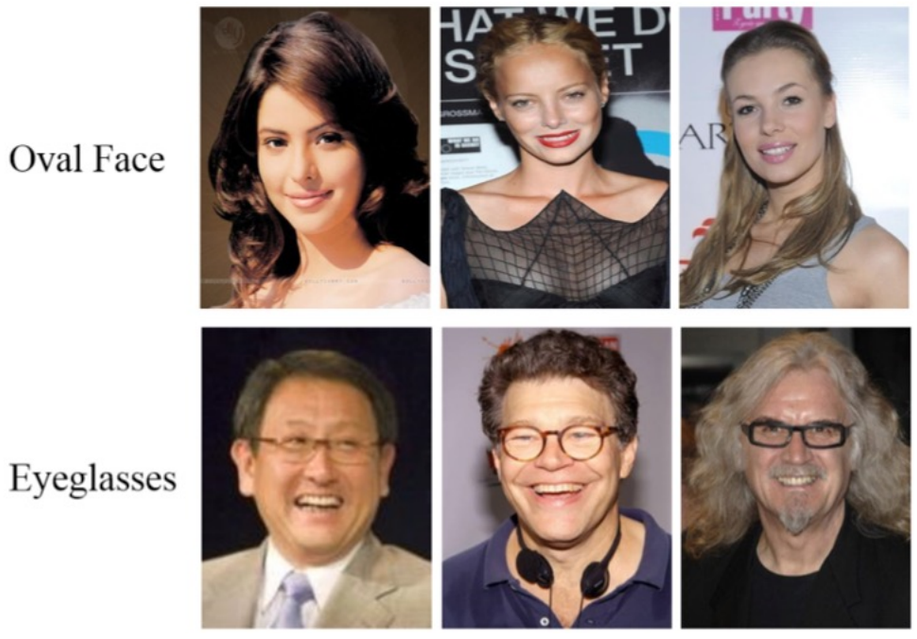
확률 분포(Probability distribution)의 추정



$P_{data}(x)$는 실제 이미지들의 정확한 분포를 가리킨다. 우리는 이 분포를 최대한 근접하게 모델링하려고 한다.

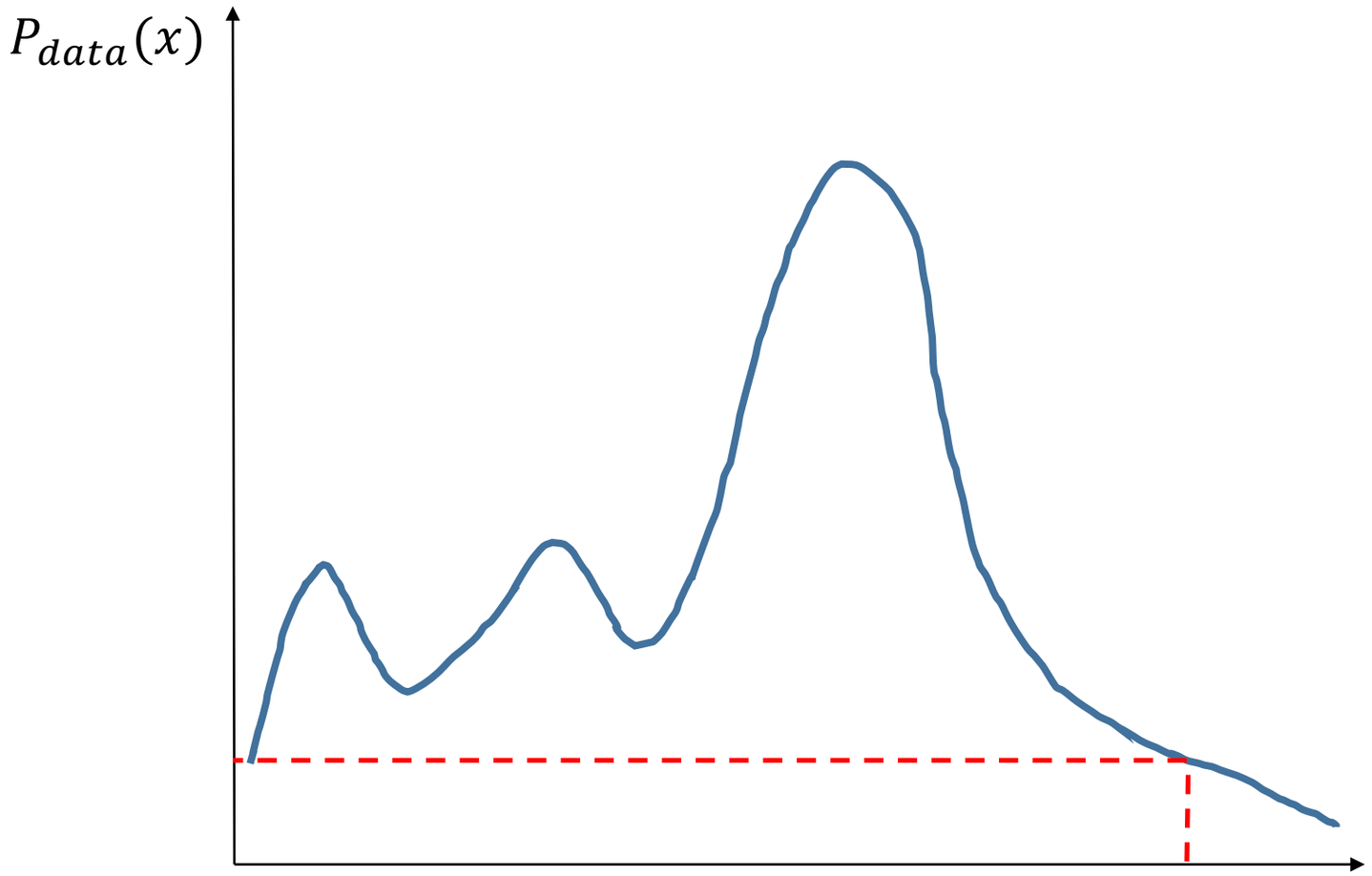
확률 분포(Probability distribution)의 추정





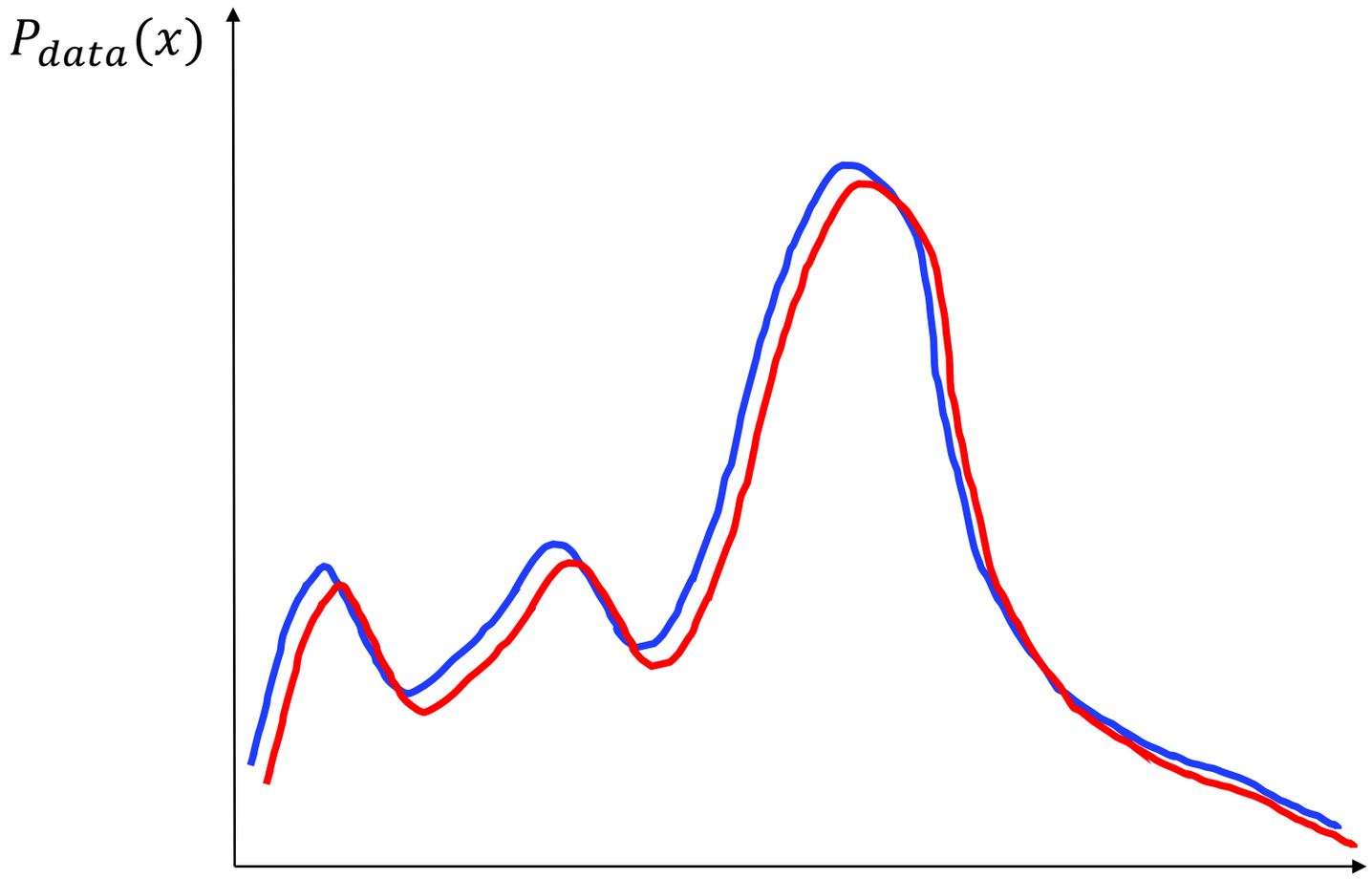
위와 같이 높은 확률을 나타내는 부분은 Oval Face 또는 Eyeglasses와 같은 특징을 나타내는 것 일수도 있다.

확률 분포(Probability distribution)의 추정



이상한 이미지를 나타내는 64x64x3 고차원 벡터일 수 있다.

확률 분포(Probability distribution)의 추정



생성모델은 $P_{data}(x)$와 가장 근사한(유사한) $P_{model}(x)$을 찾는 것이 목표이다.

$P_{data}(x)$ $P_{model}(x)$

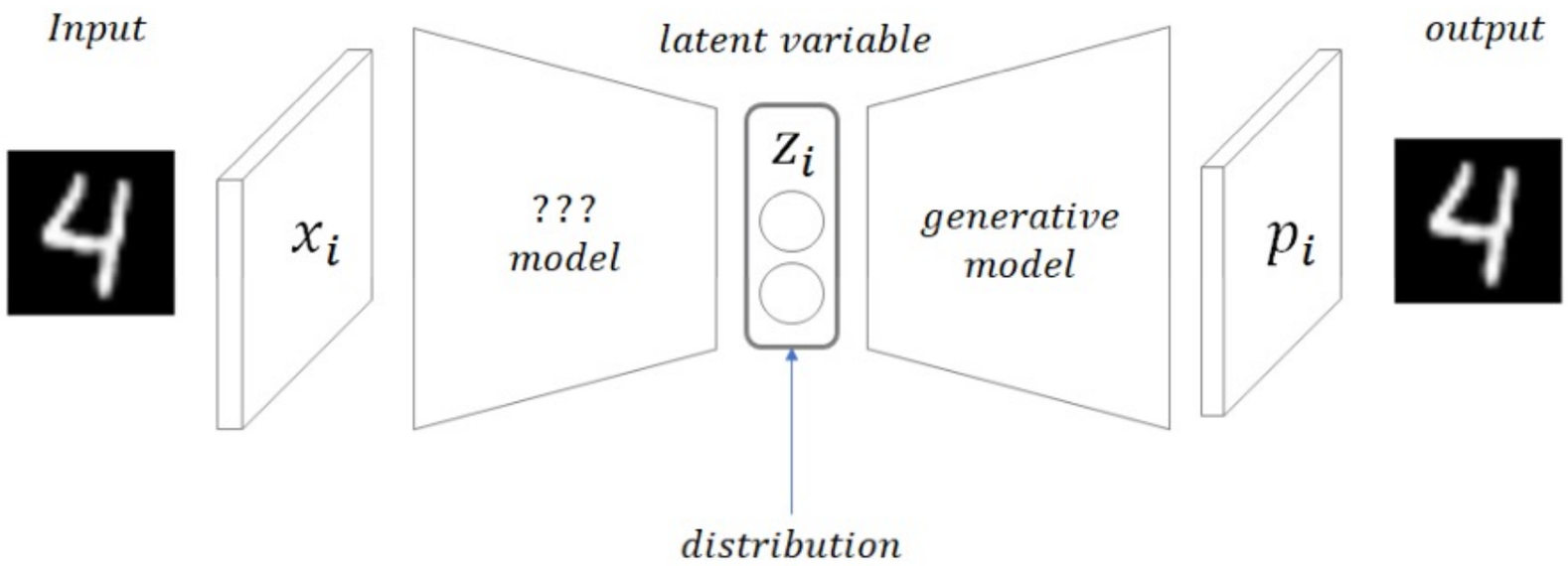
VAE의 목적

논문의 요약에 나와있는 첫 문장이다.

how can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets?

다루기 힘든 사후 분포와 큰 데이터 세트가 있는 지속적인 잠재 변수가 있는 경우, 어떻게 지시된 확률론적 모델에서 효율적인 추론과 학습을 수행할 수 있는가?

VAE의 목적

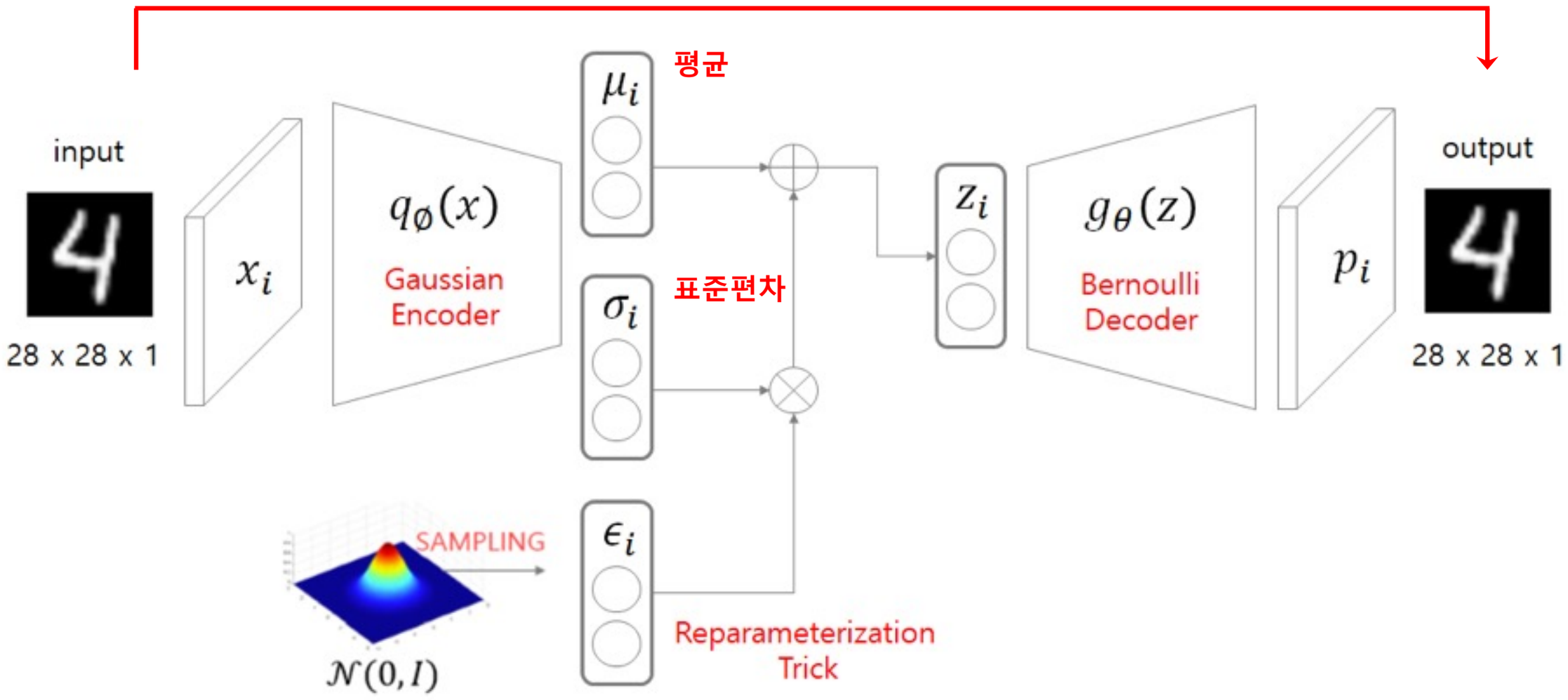


VAE의 목표는 Generative Model과 같다.
모델링 하려는 대상에서 추출된 샘플을 통해 어떤 새로운 것을 생성해 내는 것이 목표이다.

주어진 Training 데이터가 p_data(x)(확률 밀도함수)가 어떤 분포를 가지고 있다면,
샘플 모델 p_model(x) 역시 같은 분포를 갖고, 그 모델을 통해 생성된 값이 새로운
X 데이터 이길 바란다.

VAE의 구조

가장 근사(유사)하게 출력을 생성한다.

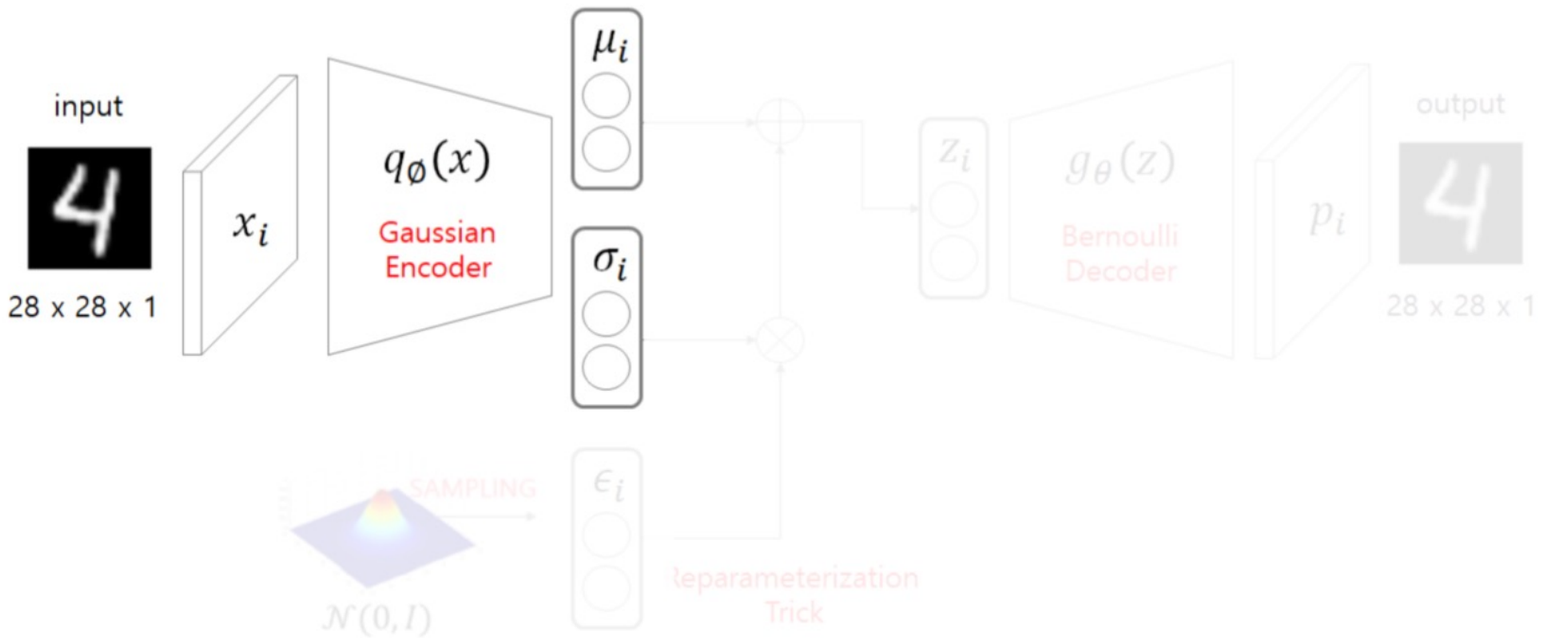


평균과 표준편차를 이용해서 정규 분포를 생성하고 Z를 샘플링한다.

Auto Encoder와 VAE는 태생적인 목적이 반대이다.

Auto Encoder는 Encoder를 위해, VAE는 Decoder를 위해 만들어졌다.

VAE –Encoder



- Input shape(x) : (28,28,1)
- $q\_\varnothing\ (x)$ 는 encoder 함수인데, x가 주어졌을때(given) z값의 분포의 평균과 분산을 아웃풋으로 내는 함수이다.
- 다시말해 q 함수(=Encoder)의 output은 μ_i,σ_i 이다.

어떤 X라는 입력을 넣어 인코더의 아웃풋은 μ_i,σ_i 이다. 어떤 데이터의 특징을(latent variable) X를 통해 추측한다. 기본적으로 여기서 나온 특징들의 분포는 정규분포를 따른다고 가정한다. 이런 특징들이 가지는 확률 분포 $q\_\varnothing\ (x)$ (정확히 말하면 $의 true 분포 (= $)를 정규분포(=Ga ussian)라 가정한다는 말이다. 따라서 latent space의 latent variable 값들은 $q_\varnothing\ (x)$의 true 분포를 approximate하는 μ_i,σ_i를 나타낸다.

> *Encoder 함수의 output은 latent variable의 분포의 μ 와 σ 를 내고, 이 output값을 표현하는 확률밀도함수를 생각해볼 수 있다.*

VAE –Encoder

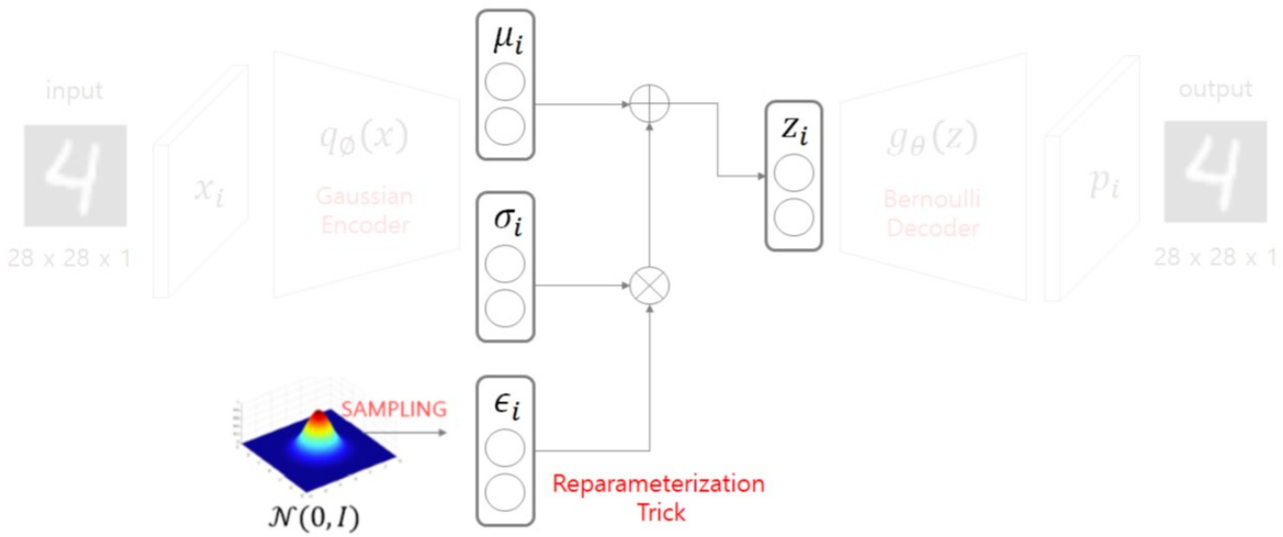
```
img_shape = (28,28,1)
batch_size = 16
latent_dim = 2


input_img = keras.Input(shape = img_shape)
x = layers.Conv2D(32,3,padding='same',activation='relu')(input_img)
x = layers.Conv2D(64,3,padding='same',activation='relu',strides=(2,2))(x)
x = layers.Conv2D(64,3,padding='same',activation='relu')(x)
x = layers.Conv2D(64,3,padding='same',activation='relu')(x)

shape_before_flattening = K.int_shape(x) # return tuple of integers of shape of x
x = layers.Flatten()(x)
x = layers.Dense(32,activation='relu')(x)

z_mean = layers.Dense(latent_dim)(x)
z_log_var = layers.Dense(latent_dim)(x)
```

VAE –Encoder




```
def sampling(args):
    z_mean, z_log_var = args
    epsilon = K.random_normal(shape=(K.shape(z_mean)[0],latent_dim),mean=0., stddev=1.)
    return z_mean + K.exp(z_log_var) * epsilon


z = layers.Lambda(sampling)([z_mean, z_log_var])
```

만약 Encoder 결과에서 나온 값을 활용해 decoding 하는데 sampling 하지 않는다면 어떤 일이 벌어질까? 당연히 는 한 값을 가지므로 그에 대한 decoder(NN)역시 한 값만 뱉는다. 그렇게 된다면 어떤 한 variable은 무조건 똑같은 한 값의 output을 가지게 된다.

하지만 Generative Model, VAE가 하고 싶은 것은, 어떤 data의 true 분포가 있으면 그 분포에서 하나를 뽑아 기존 DB에 있지 않은 새로운 data를 생성하고 싶다. 따라서 우리는 필연적으로 그 데이터의 확률분포와 같은 분포에서 하나를 뽑는 sampling을 해야한다. 하지만 그냥 sampling 한다면 sampling 한 값들을 backpropagation 할 수 없다.(아래의 그림을 보면 직관적으로 이해할 수 있다) 이를 해결하기 위해 reparmeterization trick을 사용한다.

VAE – Reparameterization Trick

$$KL(N(\mu_x, \sigma_x)||N(0, I))$$

x

Encoder

μ_x

σ_x

미분 불가

Sample z from N(μ_x, σ_x)

Decoder

$d(z)$

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

VAE – Reparameterization Trick

$$KL(N(\mu_x, \sigma_x)||N(0, I))$$

x

Encoder

b

μ_x + 미분 가능

σ_x × Sample ∈ from N(0, I)

w

$y = wx + b$의 형태

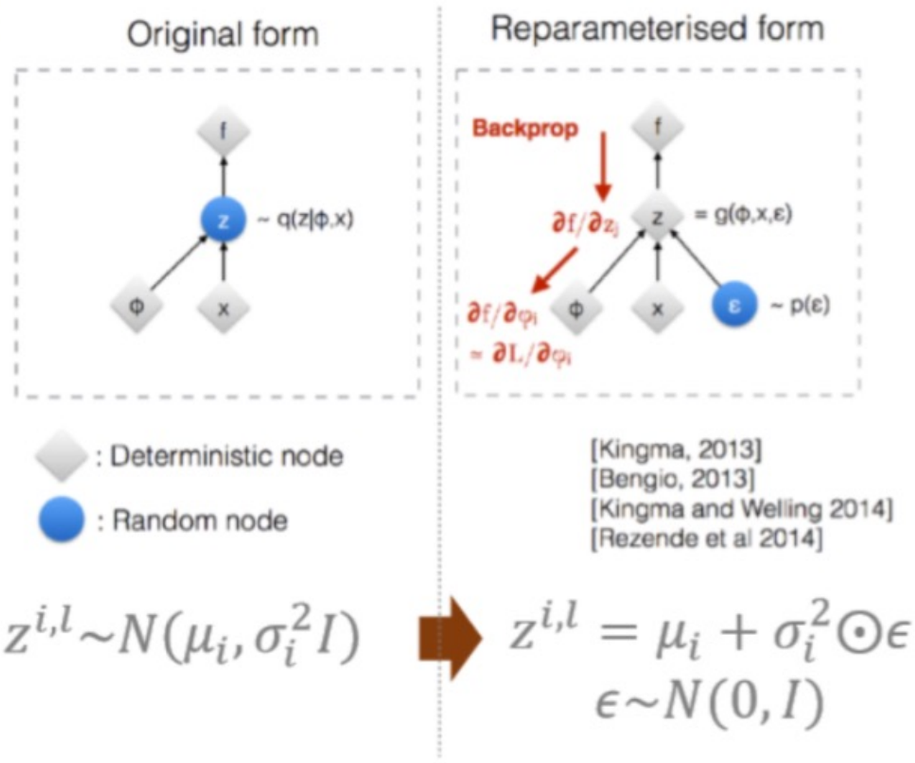
$d(z)$

Decoder

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

VAE – Reparameterization Trick

Reparameterization Trick



[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

Sampling Process

$$z^{i,l} \sim N(\mu_i, \sigma_i^2 I) \quad \Rightarrow \quad z^{i,l} = \mu_i + \sigma_i^2 \odot \epsilon, \quad \epsilon \sim N(0, I)$$

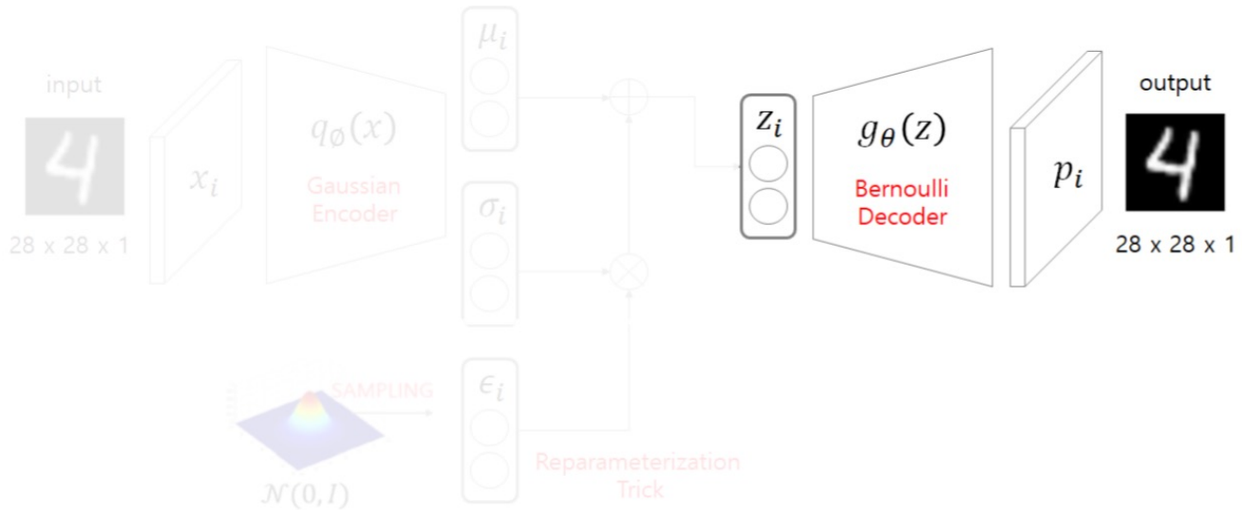
Same distribution!
But it makes backpropagation possible!!

https://home.zhaw.ch/~dueo/bbs/files/vae.pdf

정규분포에서 z1를 샘플링하는 것이나, 입실론을 정규분포(자세히는 N(0,1))에서 샘플링하고 그 값을 분산과 곱하고 평균을 더해 z2를 만들거나 두 z1,z2 는 같은 분포를 가지기 때문이다. 그래서 코드에서 `epsilon`을 먼저 정규분포에서 random하게 뽑고, 그 epsilon을 exp(z_log_var)과 곱하고 z_mean을 더한다. 그렇게 형성된 값이 z가 된다.

> *latent variable에서 sample된 z라는 value (= decoder input)이 만들어진다.*

VAE –Decoder




```
## 8.25 VAE decoder network, mapping latent space points to imgaes

decoder_input = layers.Input(K.int_shape(z)[1:])
x = layers.Dense(np.prod(shape_before_flattening[1:]),activation='relu')(decoder_input)
x = layers.Reshape(shape_before_flattening[1:])(x)
x = layers.Conv2DTranspose(32,3,padding='same',activation='relu',strides=(2,2))(x)
x = layers.Conv2D(1,3,padding='same',activation='sigmoid')(x)

decoder = Model(decoder_input, x)
z_decoded = decoder(z)
```

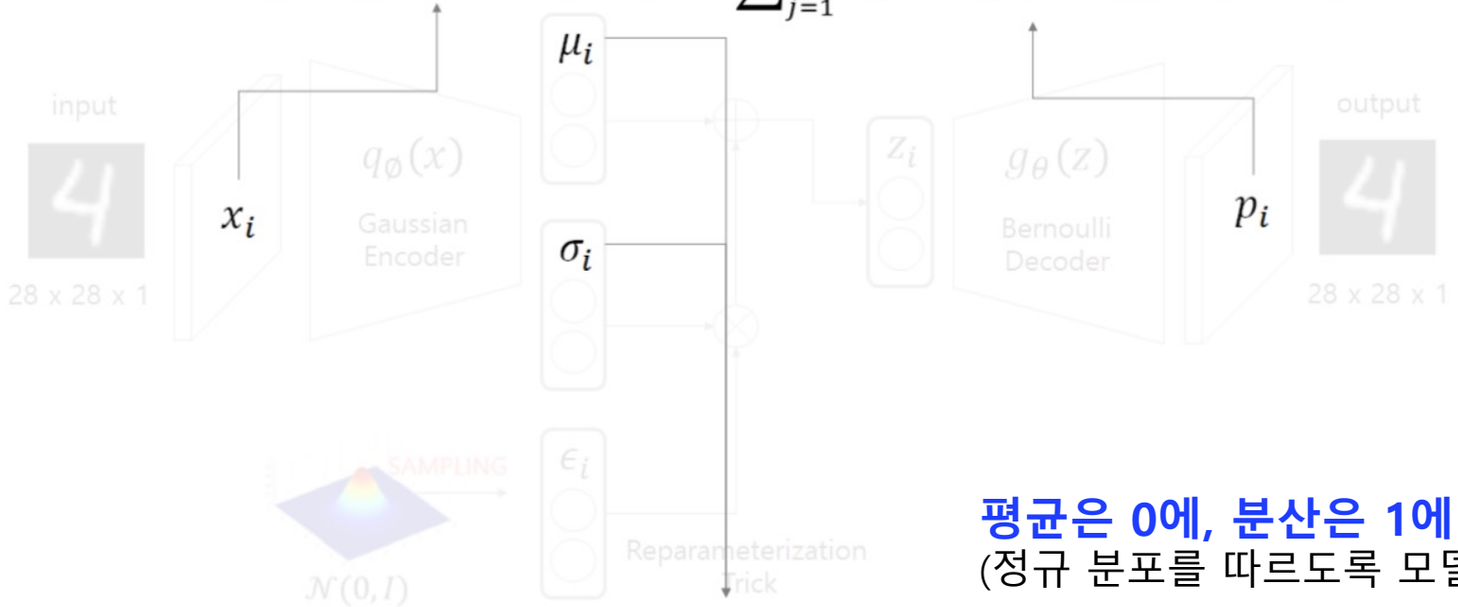
z 값을 g 함수(decoder)에 넣고 deconv(코드에서는 Conv2DTranspose)를 해 원래 이미지 사이즈의 아웃풋 z_decoded가 나오게 된다. 이때 p_data(x)의 분포를 Bernoulli 로 가정했으므로(이미지 recognition 에서 Gaussian 으로 가정할때보다 Bernoulli로 가정해야 의미상 그리고 결과상 더 적절했기 때문) output 값은 0~1 사이 값을 가져야하고, 이를 위해 activatino function을 sigmoid로 설정해주었다. (Gaussian 분포를 따른다고 가정하고 푼다면 아래 loss를 다르게 설정해야한다.)

VAE – Loss function

Loss는 크게 2가지 부분이 있다.

입력과 출력이 얼마나 유사한가?

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



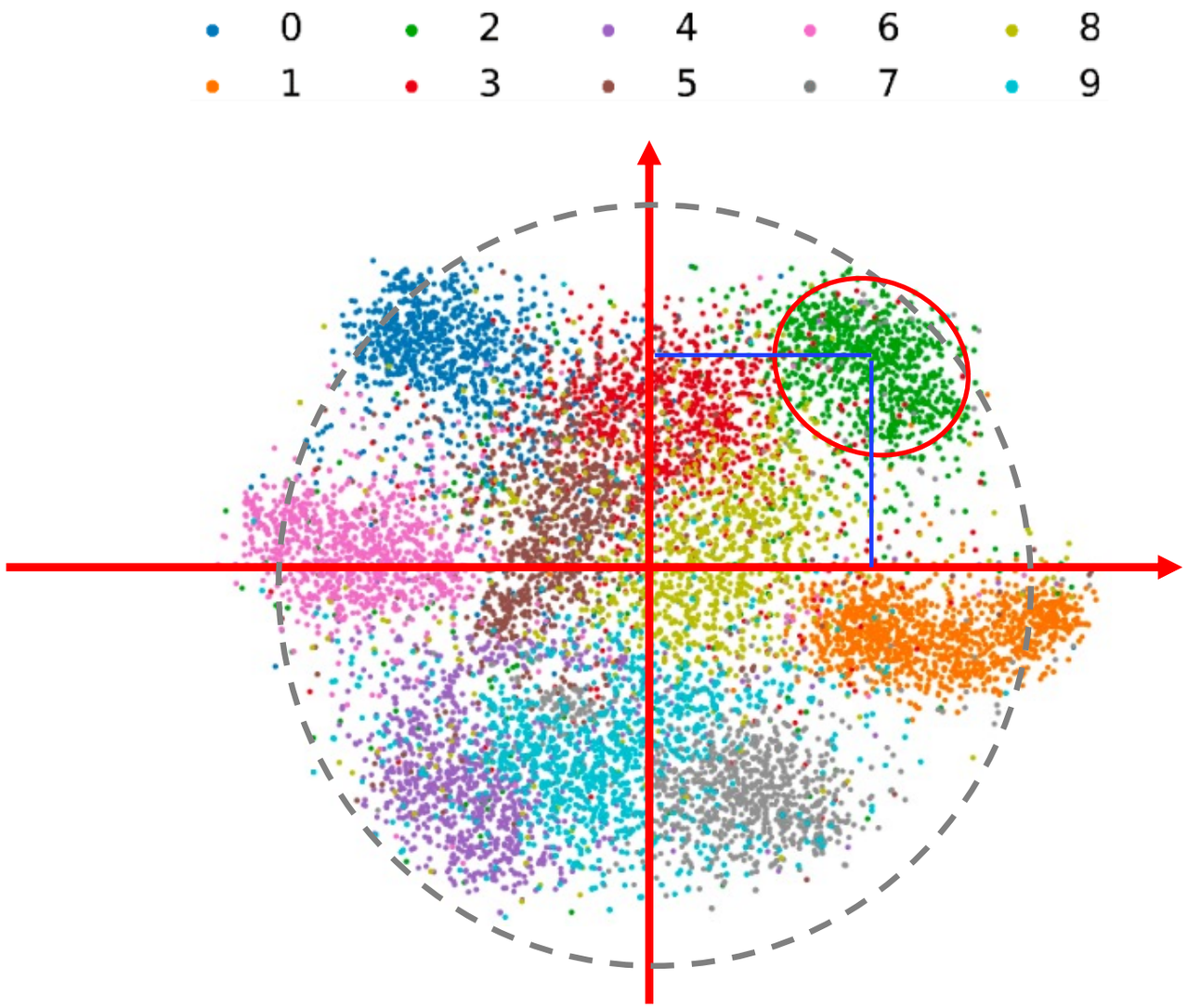
평균은 0에, 분산은 1에 가깝게 나타나야 한다.
(정규 분포를 따르도록 모델링한다.)

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

```
def vae_loss(self, x, z_decoded):
        x = K.flatten(x)
        z_decoded = K.flatten(z_decoded)
        xent_loss = keras.metrics.binary_crossentropy(x,z_decoded)
        kl_loss   = -5e-4*K.mean(1+z_log_var-K.square(z_mean)-K.exp(z_log_var),axis=-1)
        return K.mean(xent_loss + kl_loss)
```

- Reconstruction Loss(code에서는 xent_loss)
- Regularization Loss(code에서는 kl_loss)

MNIST의 VAE 벡터z의 2d 임베딩 시각화



각 Digit별로 평균과 분산이
다르게 분포된 것을 볼 수 있다.

VAE의 특성 정리

Simple & Stable : 학습 프로세스가 간단하면서 안정적이다.

Pure generation : 순수 생성을 위한 목적으로 사용될 수 있다.

Blurry : GAN에 비해 상대적으로 흐릿한 이미지를 생성함.

참고 서적 및 github :미술관에간 딥러닝 프로젝트

https://github.com/rickiepark/GDL_code