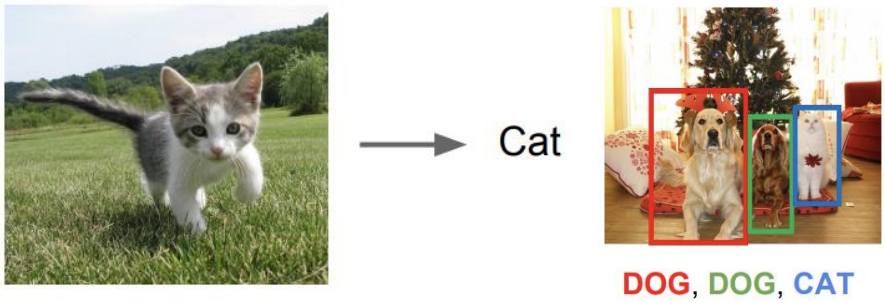
Supervised Learning vs. Unsupervised Learning

Supervised 일명 **지도학습**은, 학습데이터의 label, 정답이 주어진 학습법.

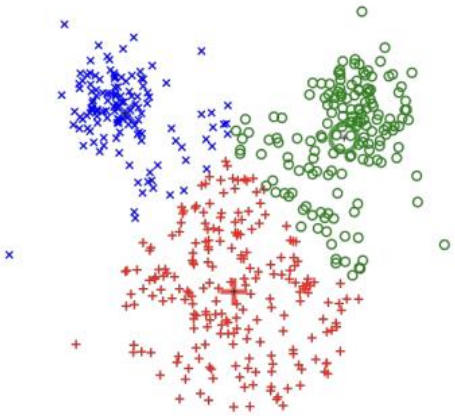
아래 그림과 같이 고양이 사진에 'cat'이라는 label 이 주어지고, 또 여러 물체가 있을때 어떤 물체가 각각 어떤 클래스에 속하는지 label 이 주어지기도 합니다.



이처럼 주어진 데이터와 그 label 을 학습하며 새로운 data 와 그에 따른 label 을 맵핑하는 함수를 찾는데 지도학습이라고 할 수 있습니다.

반대로 **Unsupervised Learning**, **비지도 학습**은 정답 레이블이 주어지지 않고 데이터를 잘 포착해 결과물을 도출하는 학습방법입니다.

Ex) 머신러닝의 K-means clustering

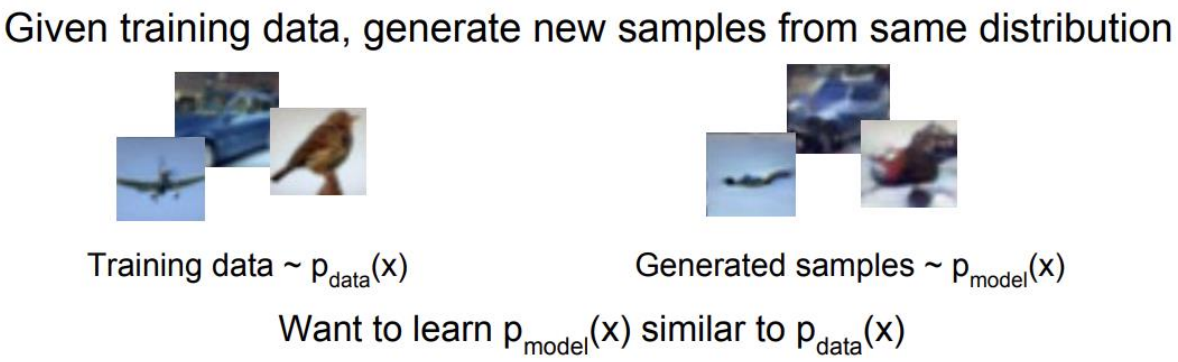


label 이 없기 때문에, data 의 **hidden structure** 를 찾아내는게 목표입니다.

종종 책이나 강의에서 지도학습은 이끌어주는 선생님이 있고,
비지도 학습은 혼자 배워나가는 독학과도 같은 느낌이다. 라고 소개가 되어 있는데요

지도학습이 좀 더 쉽기 때문에(?) 비지도 학습은 그에비해 아직 미개척 학문인 느낌이 큽니다.
또한 data 에 label 을 일일히 붙이는 작업이 필요없고, 그저 데이터만 있으면 바로 사용 가능하기 때문에 data 가 cheap 하다는 장점.

Generative Models



레퍼런스가 될 training data 가 주어졌을때 그와 비슷한 분포를 가지는 new data 를 생성하는게 생성모델의 역할입니다.
generative model 이 전부 새로운 data 를 생성하는 역할을 가지지만, 굵직굵직한 특성에 따라 아래와 같이 분리될 수 있습니다.

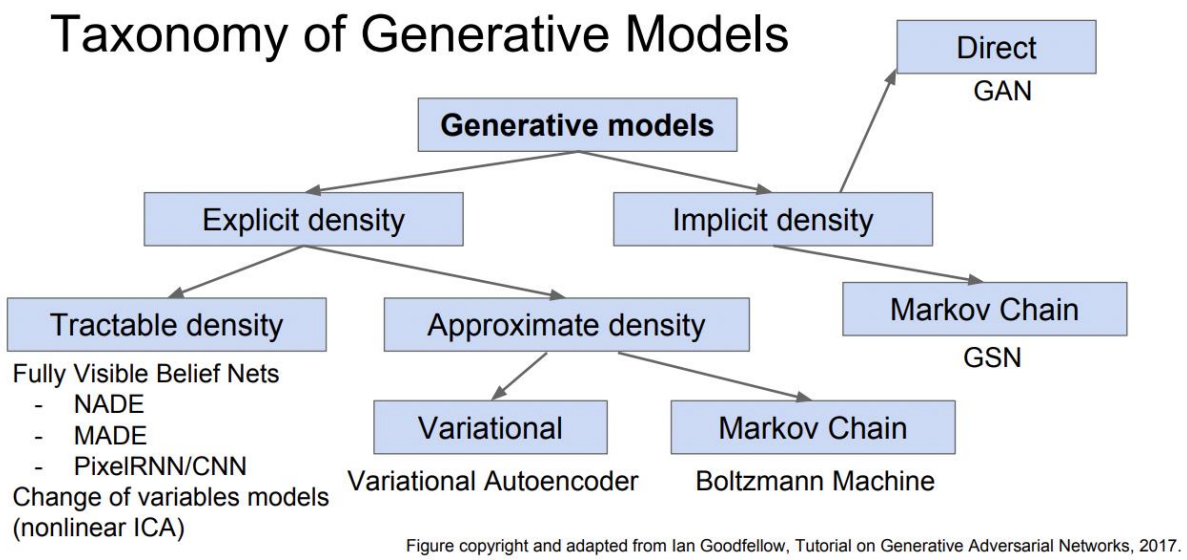


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

먼저, pdata(x)pdata(x)(실제 데이터 분포)에 근사하고자 하는 pmodel(x)pmodel(x)을 어떻게 정의하느냐에 따라 아래와 같은 큰 갈래로 나뉩니다.

Explicit density: pmodel(x)pmodel(x)이 어떤 분포를 띄는지를 정의하고 찾는데 초점을 둡니다.
Explicit density 모델은 training data 의 likelihood 를 높이는 방향으로 학습을 합니다. x1 xix1 xi 까지가 각 pixel 이 등장할 확률이라면, 해당 pixel 들로 구성된 이미지가 나타날 확률은 각 pixel 들의 확률곱입니다. 따라서 아래와 같은 식으로 나타낼 수 있습니다. Loss 도 계산할 수 있어서 학습 정도를 알 수 있습니다.

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

Likelihood of image x

Probability of i'th pixel value given all previous pixels

위에 장점만 보면 loss 도 계산할 수 있는 explicit model 이 훨씬 유리한 방법같아보일 수 있지만, 현재는 생성모델 중 GAN(implicit)이 제일 잘나가고 있습니다. 그 이유는 바로 모델을 정의하는게 한계가 있기 때문입니다. 데이터가 더 복잡해질수록 분포를 식으로 표현해서 계산하기 어렵기 때문에 implicit model 쪽을 많이 택합니다.

Implicit density:
pmodel(x)pmodel(x)이 어떤 분포를 띄는지 정의하는데는 관심이 없고, 단지 sample 을 생성할 수 있는 수준을 원합니다. pmodel(x)pmodel(x)을 sampler 로 사용.

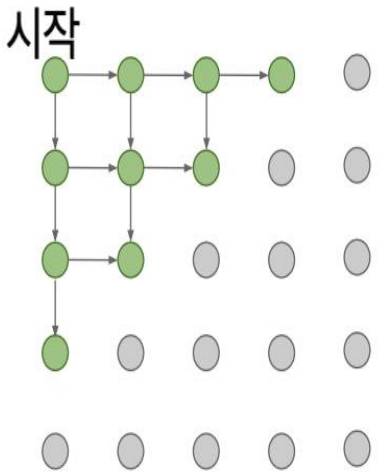
지금은 무슨 차이인지 와닿지 않을 수 있습니다. 뒤에 나올 GAN 의 내용을 보면 이해가 되겠습니다.

Pixel RNN - tractable density
Pixel RNN 은 왼쪽 위 코너의 시작점으로 부터 상하좌우로 뻗어나가면서 이미지를 pixel by pixel 로 생성하는 방법입니다.

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

Drawback: sequential generation is slow!



이때 새로 만들어지는 픽셀은 인접한 픽셀들의 영향을 받아서 새로 생성됩니다. 이전 결과에 영향을 받는 구조에는 RNN 이 적합하기 때문에, 이전 픽셀들에 대한 dependency 는 LSTM 같은 RNN 등으로 표현이됩니다.

- : 매우 느리다. feed-forward process 같이 레이어를 몇번 거치면 뽕하고 나타나는게 아니라. 작업을 모든 픽셀에 대해 순차적,반복적으로 해야 된다.

Pixel CNN

Still generate image pixels starting from corner

Dependency on previous pixels now modeled using a CNN over context region

Training is faster than PixelRNN
(can parallelize convolutions since context region values known from training images)

Generation must still proceed sequentially
=> still slow

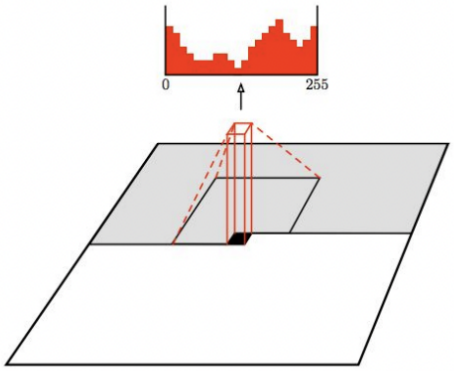


Figure copyright van den Oord et al., 2016. Reproduced with permission. 회색 : 아마 생성된 context region

Pixel RNN 의 에서 RNN 을 CNN 으로 대체한 방법이 Pixel CNN 입니다.

Pixel RNN 처럼 이미지의 한쪽 끝에서 시작하지만, 이미지 생성에 영향을 주는 인접한 좌표들에 한꺼번에 CNN 을 하는 방식으로, Pixel RNN 보다 빠르다는 장점이 있습니다.

위에서 본 방법은 ecplixit-tractable density function 이었습니다.
지금부터 볼 VAE 는 intractable density function 입니다. (복잡해서 계산을 할 수 없다)

VAEs define intractable density function with latent **z**:

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

계산을 해서 직접 optimixe 할 수 없기 때문에 function 의 하한선을 찾아서 그 하한선을 maximize 하는 방법으로 최적화를 대신합니다.
VAE 의 background 가 되는 AE 를 먼저 살펴보겠습니다.

Auto Encoder
: Unsupervised approch to learn lower dimensional feature representtion from unlabeled data
: 레이블이 없는 데이터에서 feature representation 을 뽑는 비지도 학습법

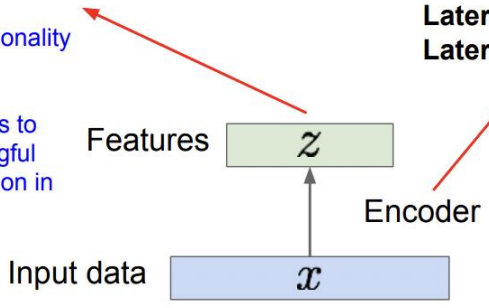
z: feature vector

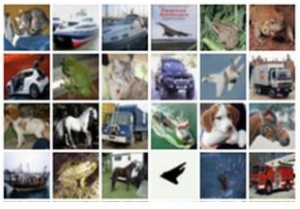
z usually smaller than **x**
(dimensionality reduction)

Q: Why dimensionality reduction?

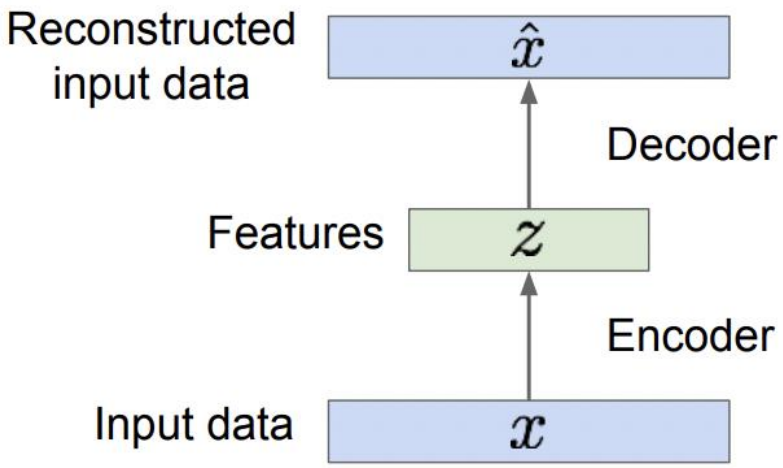
A: Want features to capture meaningful factors of variation in data

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN

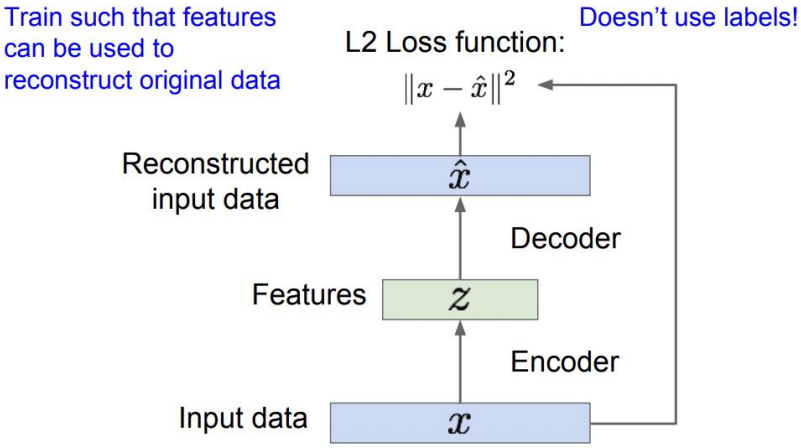




1. input data x 에서 feature vector z 를 추출한다, downsample : **Encoder**
- x 에서 의미있는 요소를 추출한게 z 이기 때문에 대체적으로 z 의 dimension 이 x 보다 작다.



2. feature z 에서 Recondtructed input data ^xx^를 다시 만들어낸다, upsample : **Decoder**

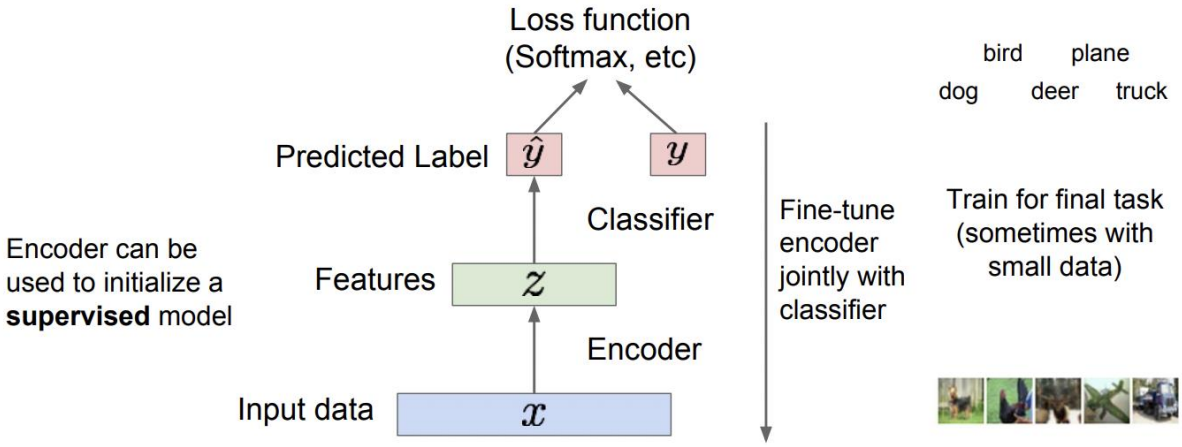


Auto-Encoder 는 1 번으로 z 를 생성하고, 2 번으로 ^xx^를 생성해서 최종적으로

xx 와 ^xx^차이를 최대한 줄이도록 feature z 를 학습합니다.

no labels! xx 와 ^xx^만 필요!!

Decoder 는 사실상 주요기능을 하지 않습니다. z 를 학습하는데 있어서 input x 와 비교할 기준이 필요하기때문에, 그 기준이 되는 ^xx^을 생성하는데 사용되는 도구일뿐. 따라서 학습후 그냥 버려지게 됩니다.



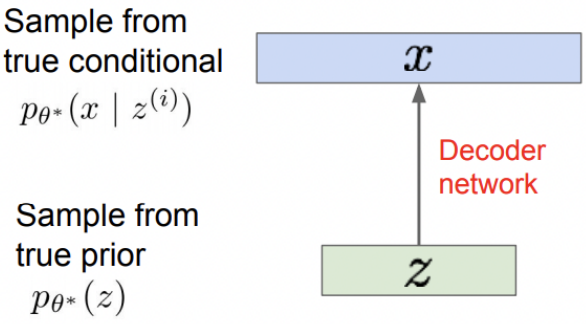
Decoder 를 제거한 모델은 이제 feature 만 남았습니다.
이 feature 는 적절한 학습을 통해 input data x 의 중요한 feature 를 추출해 낼 수 있게 되었죠.
사실상 이 feature 를 data 의 특성을 잘 반영하게끔 추출해내는게 Auto Encoder 의 목적입니다. Decoder 는 도구였을 뿐이죠.

이제 이 feature 는 supervised model 의 input 으로 들어가, classification 하는데 사용합니다. (Generative model 이 아님!)

기존 classification 과 비교했을때, 왜 이렇게 복잡한 거치는지에 대한 이유는 아래와 같습니다.
data 가 넉넉하지 않을때, overfit/underfit 되는걸 최대한 줄이기 위해 이런 방법을 사용합니다.

VAE

Auto-Encoder 가 잘 추출한 feature 를 사용해 이미지 클래스를 분류했다면, VAE 는 이 feature 로 새로운 이미지를 생성할 수는 없을까?하는 의문에서 출발합니다.



여기 vector z 가 있습니다. 그리고 x 는 이 생성모델에 latent vector z 와 parameter θθ 를 집어넣은 결과입니다.

- zz: latent vector. Gaussian 분포같은 랜덤 노이즈가 들어가기도 함. (Approximation)
- pθ∗(z)pθ∗(z): parameter 가 θθ 일때, latent vector z 를 sampling 할 수 있는 확률밀도함수
- pθ(x|z)pθ(x|z): parameter 가 θθ 이면서, z 가 주어졌을 때 x 를 생성해내는 확률밀도함수

여기서 θθ 를 실제 분포와 가깝게 찾는것이 목표입니다. 따라서 p(z)에서 p(x|z)를 만드는 Decoder network 를 복잡한 구조도 핸들링 가능한 neural network 로 구성을 하고, 아래와 같은

pθ(x)pθ(x) - paremeter 가 θθ 일때 x 가 나올 likelihood 를 최대화 시키는 방향으로 학습을 합니다.

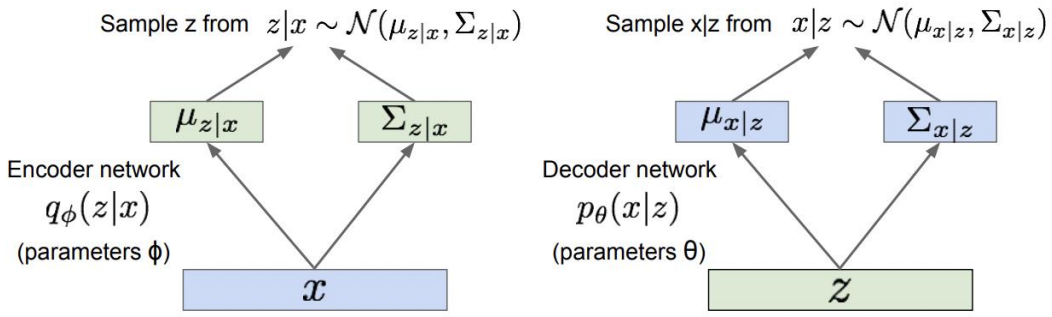
$$p\theta(x) = \int p\theta(z)p\theta(x|z)dz$$

하지만 모든 z 에 대해 p(x|z)의 적분을 취해줄 수 없다는 문제점이 있습니다. 이게 바로 intractable 한 문제입니다. 계산이 불가능하다는 뜻입니다. 그래서 차용한 방식이 VAE 입니다.

Posterior density also intractable: $p_\theta(z|x) = p_\theta(x|z)p_\theta(z)/p_\theta(x)$

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

위에서는 decoder network 만 있었다면 여기서는 encoder netowrk q 를 추가합니다.
여기서 qϕ(z|x)qϕ(z|x)는 pθ(z|x)pθ(z|x)를 근사하는 encoder network 입니다.



Decoder Encoder 구조로 구성된 VAE 네트워크의 구조입니다.
왼쪽은 Encoder, 오른쪽은 Decoder 네트워크입니다.

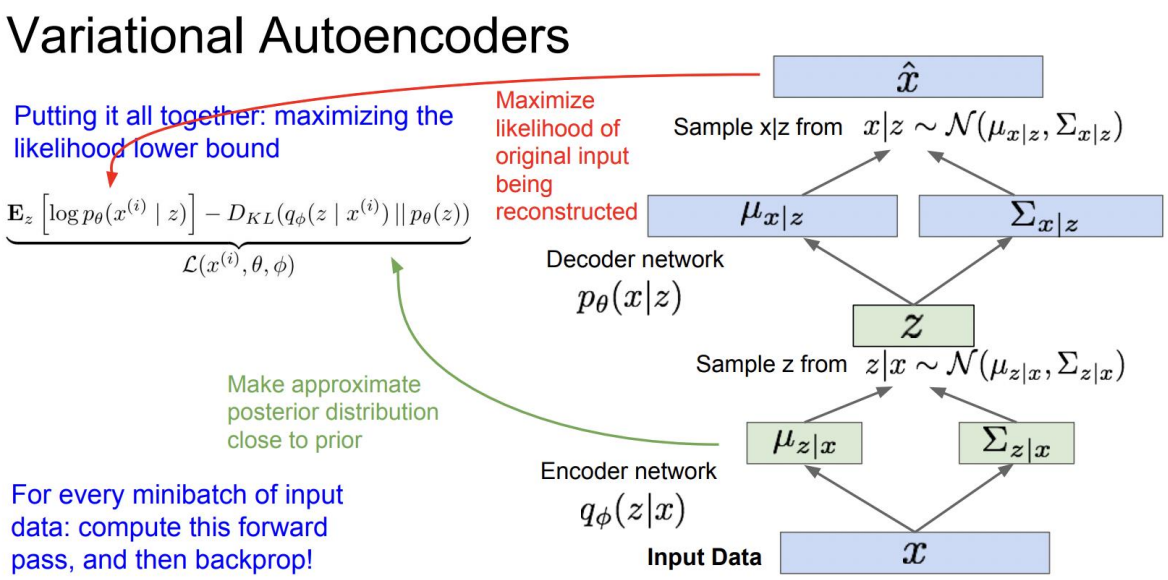
Encoder

- Encoder qϕ(z|x)qϕ(z|x) : x 를 input 으로 받아서 mean,covariance 추출 후, z space 상에서 분포를 생성.
- z 는 gaussian 분포를 따른다고 가정.(예시일뿐, 다른 분포도 가능)

Decoder

- gaussian 분포로부터 z 를 sampling.
- sampling 한 z 를 가지고 decoder pθ(z|x)pθ(z|x)는 x space 상의 확률분포를 생성하고, x 를 이 분포로부터 sampling

이러한 Encoder-Decoder x-> z ->x 구조를 가지기 때문에 Auto-Encoder 라고 할 수 있고, 결과적으로 유의미한 feature vector z 를 얻을 수 있습니다.



GAN (implicit density)

Generative Adversarial networks : 모델을 직접 optimize 할 수 없다. optimize 는 포기하고 sampler 의 기능을 극대화시키는 방법