Generative Models

Unsupervised Learning

1. Background: Supervised Learning
    a. Given data (x, y)
        i. x: data
        ii. y: label
    b. Learn a function to map x → y
    c. Examples
        i. classification, regression, object detection, semantic segmentation, image captioning, etc
2. Unsupervised Learning
    a. Given data x
    b. Learn some underlying hidden structure of the data
    c. Examples
        i. clustering, dimensionality reduction, feature learning, density estimation, etc

Generative Models

1. Generative models
    a. Given training data
    b. Generate new samples from same distribution
        i. want to learn $p_{model}(x)$ similar to $p_{data}(x)$
            - $p_{data}(x)$ ~ training data distribution
            - $p_{model}(x)$ ~ generated sample distribution
2. Density estimation
    a. 관측된 데이터들의 분포로부터 원래 변수의 확률 분포 특성을 추정하는 것
        i. 어떤 변수 x의 밀도 (density)를 추정하는 것은 x의 확률 밀도 함수 (probability density function)를 추정하는 것과 동일함

b. For supervised

    i. 미리 확률 밀도 함수를 정해두고 데이터들로부터 모델의 파라미터만 추정하는 방식 (parametric)

c. For unsupervised

    i. 어떠한 사전 정보나 지식 없이 순수하게 관측된 데이터만으로 확률 밀도 함수를 추정하는 방식 (non-parametric)

d. Density estimation is a core problem in unsupervised learning

    i. generative model addresses density estimation

e. Type

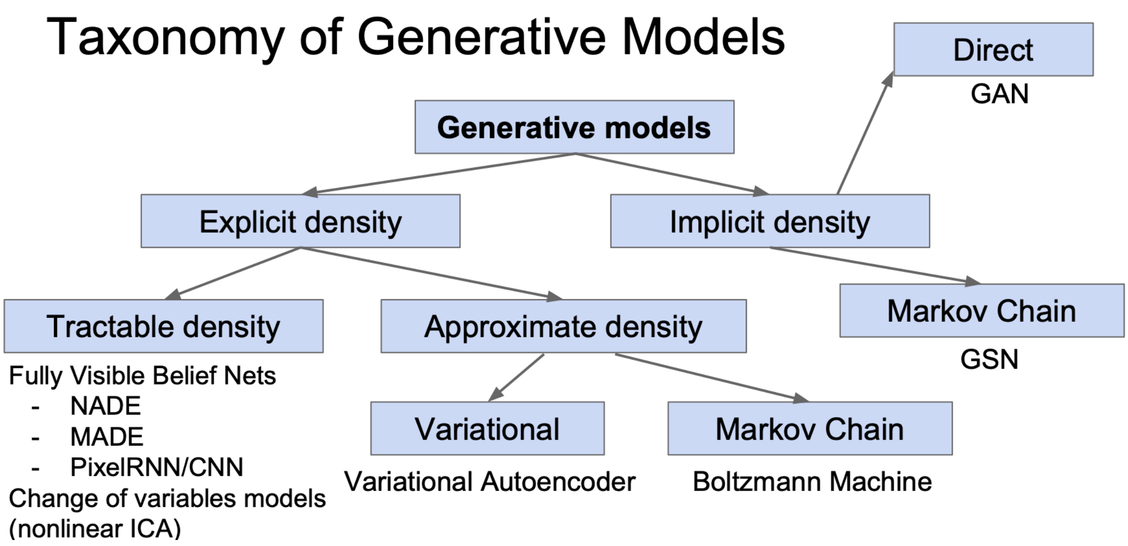
    i. explicit density estimation

        - explicitly define and solve for $p_{model}(x)$

    ii. implicit density estimation

        - learn model that can sample from $p_{model}(x)$ without explicitly defining it

3. Taxonomy

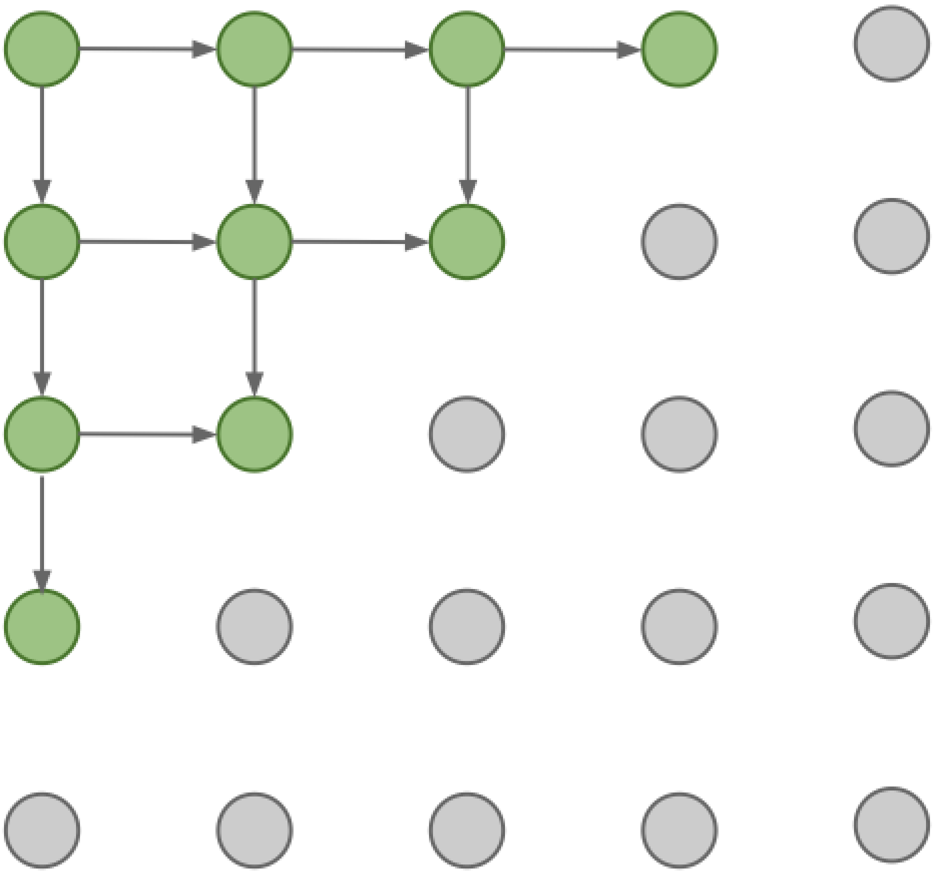


PixelRNN & PixelCNN

1. PixelRNN

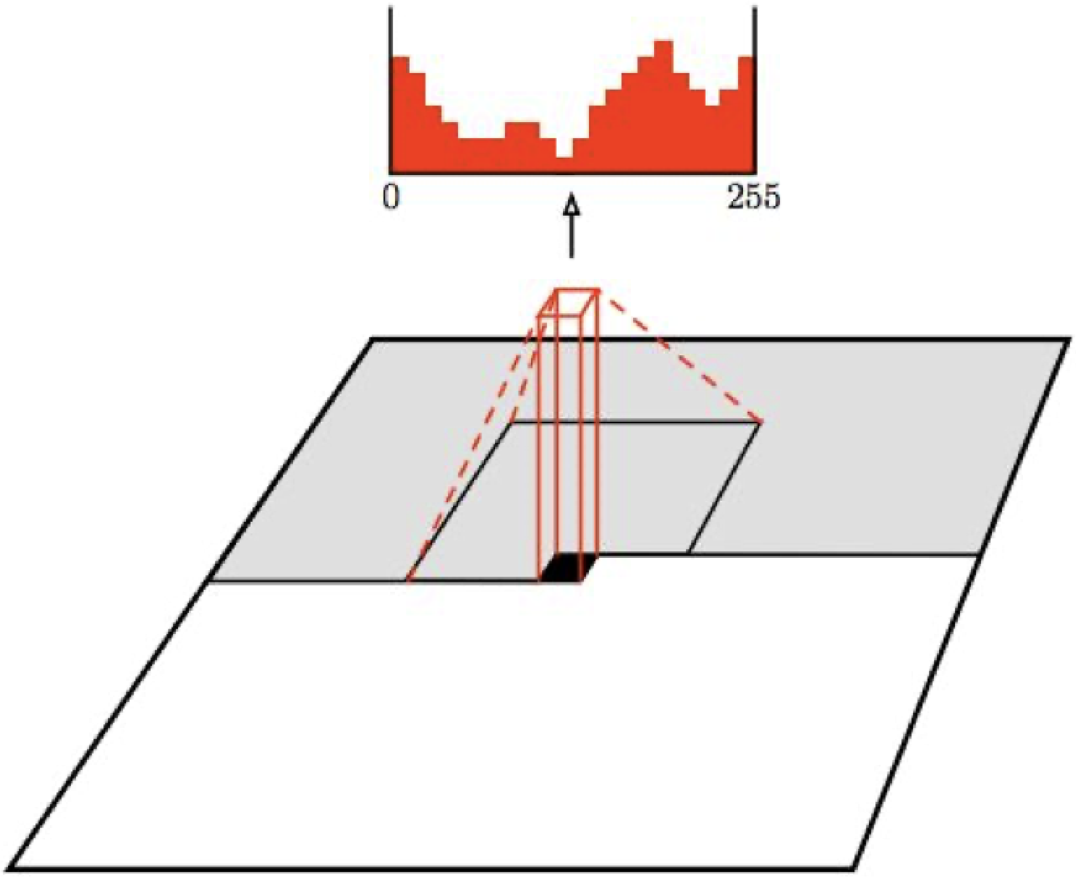
    a. Generate image pixels starting from corner

    b. Dependency on previous pixels modeled using an RNN (LSTM)



c. Drawback
    i. sequential generation is slow b/c iteratively go through and add all the pixels

2. PixelCNN
    a. Still generate image pixels starting from corner
    b. Dependency on previous pixels now modeled using a CNN over context region
        i. train by maximizing the likelihood of training images



c. Advantage

  i. training is faster than PixelRNN b/c can parallelize convolutions since context region values known from training images

d. Drawback

  i. generation must still proceed sequentially, so still slow

3. Pros & Cons

  a. Pros

    i. can explicitly compute likelihood $p(x)$

    ii. explicit likelihood of training data gives good evaluation metric

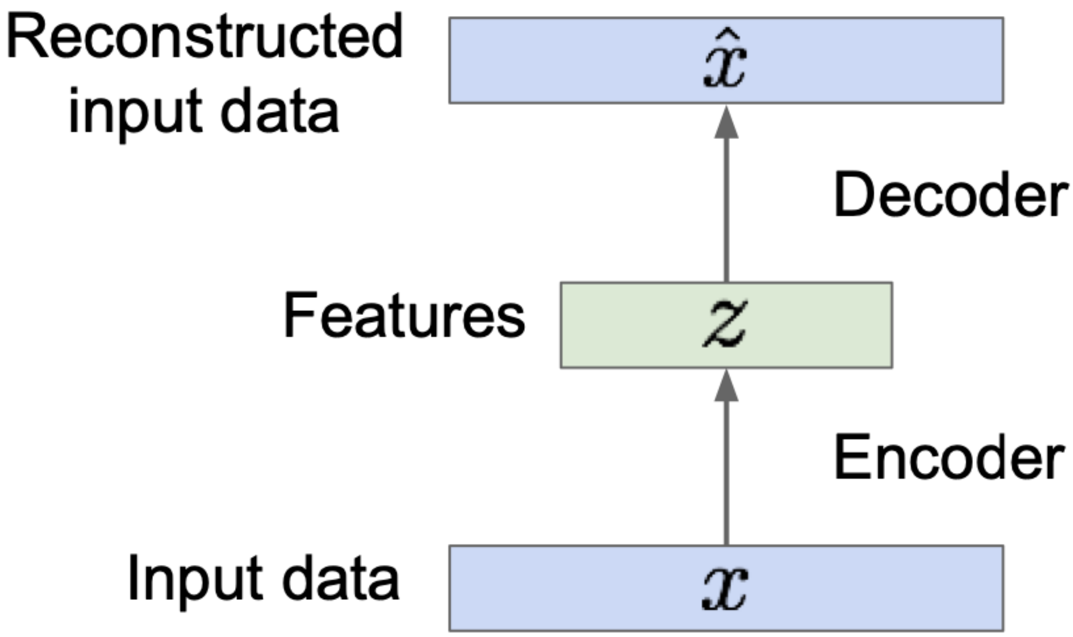
    iii. good samples

  b. Cons

    i. sequential generation is too slow

Variational Autoencoders (VAE)

1. PixelCNN vs. VAE
    a. PixelCNNs define tractable density function, optimize likelihood of training data $p_\theta(x)$
    b. VAEs define intractable density function with latent z
        i. cannot optimize directly, derive and optimize lower bound on likelihood, instead
2. Background: Autoencoder
    a. Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data



    b. Encoder
        i. generate z from x
            - z usually smaller than x b/c dimensionality reduction
                - dimensionality reduction 을 통해 중요한 feature만 z로 선정됨
    c. Decoder
        i. reconstruct $\hat{x}$ from z
    d. Process
        i. Train using L2 loss function

- $||x - \hat{x}||^2$

ii. Throw away decoder

iii. Use encoder to initialize a supervised model

- softmax loss function and fine-tuning

e. Purpose

i. Autoencoder의 decoder 부분은 encoder를 학습시키는 것이 목적

ii. 데이터 생성이 목적이 아님

iii. 데이터를 잘 압축하는 것 (i.e., 데이터의 특징을 잘 추출하는 것, 데이터의 차원을 잘 줄이는 것) 이 목적

3. Variational autoencoder

a. Purpose

i. VAE는 decoder를 학습시키기 위해 encoder를 사용함

ii. decoder를 이용하여 데이터를 생성하는 것이 목적

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

i. z is sampled from true prior $p_{\theta^*}(z)$

ii. x is sampled from true conditional $p_{\theta^*}(x|z^{(i)})$

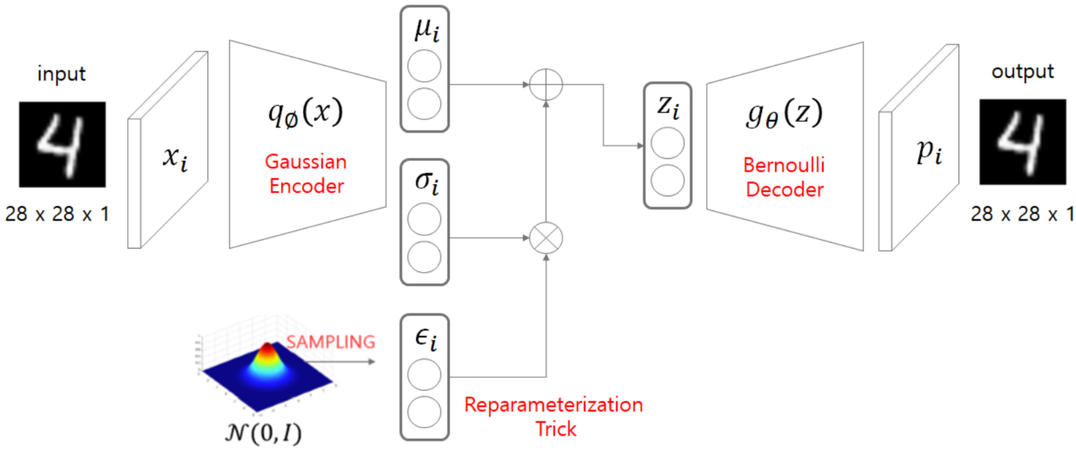
c. Estimate the true parameters θ^* of the generative model

d. How?

i. choose prior $p(z)$ to be simple, e.g., Gaussian

ii. conditional $p(x|z)$ is complex (generates image), so present with neural network

e. Overall structure



f. Encoder

i. 데이터 x를 입력으로 넣어서 데이터의 특징을 추측함

ii. 이 때, 특징들의 분포는 정규 분포를 따른다고 가정함

iii. 따라서 아웃풋으로 특징 분포의 μ_i, σ_i 를 리턴함

iv. 문제점: 분포 내 데이터를 어떻게 뽑아서 사용할까?

- reparameterization trick 을 사용하여 샘플링함
  - 직접 샘플링: 가우시안 분포에서 직접 샘플링을 하면 역전파가 불가능함
  - reparameterization trick: ϵ 을 N(0, 1)에서 샘플링해서 σ_i 와 곱하고 μ_i 와 더하여 같은 확률 분포에서 값을 샘플링함

g. Decoder

i. 특징 z를 통해 데이터 x를 복원함 (z를 줬을 때 x가 나올 확률을 maximize 해야함)

ii. loss function

- reconstruction error: 복원된 샘플이 얼마나 원본과 유사하도록 만들어줌
- regularization term: 이상적인 z의 분포와 encoder를 통해 얻은 z의 분포가 유사하도록 만들어줌 (KL-divergence)

4. Pros & Cons

a. Pros

i. principled approach to generative models

ii. allows inference of q(z|x), can be useful feature representation for other tasks

b. Cons

i. not as good evaluation as PixelRNN or PixelCNN

ii. samples blurrier and lower quality compared to SOTA (GANs)

Generative Adversarial Networks (GAN)

1. Overview

   a. GANs don't work with any explicit density function

   b. Take game-theoretic approach, instead

      i. learn to generate from training distribution through 2-player game

2. GANs

   a. Problem

      i. want to sample from complex, high-dimensional training distribution

   b. Solution

      i. sample from a simple distribution, e.g., random noise

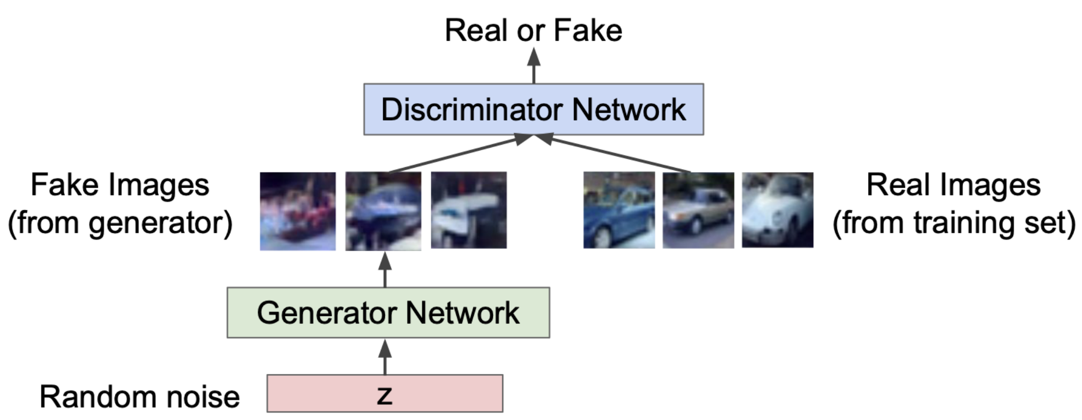
   c. Training

      i. Generator network

         - try to fool the discriminator by generating real-looking images

      ii. Discriminator network

         - try to distinguish between real and fake images



   d. After training
      i. Use generator network to generate new images
3. Pros & Cons
   a. Pros
      i. SOTA samples
   b. Cons
      i. more unstable to train
      ii. can't solve inference queries such as $p(x)$, $p(z|x)$