Lec13 Generative Models

1. Unsupervised Learning

이전까지 supervised learning을 다뤘다면 이제부터는 unsupervised learning을 다룬다.

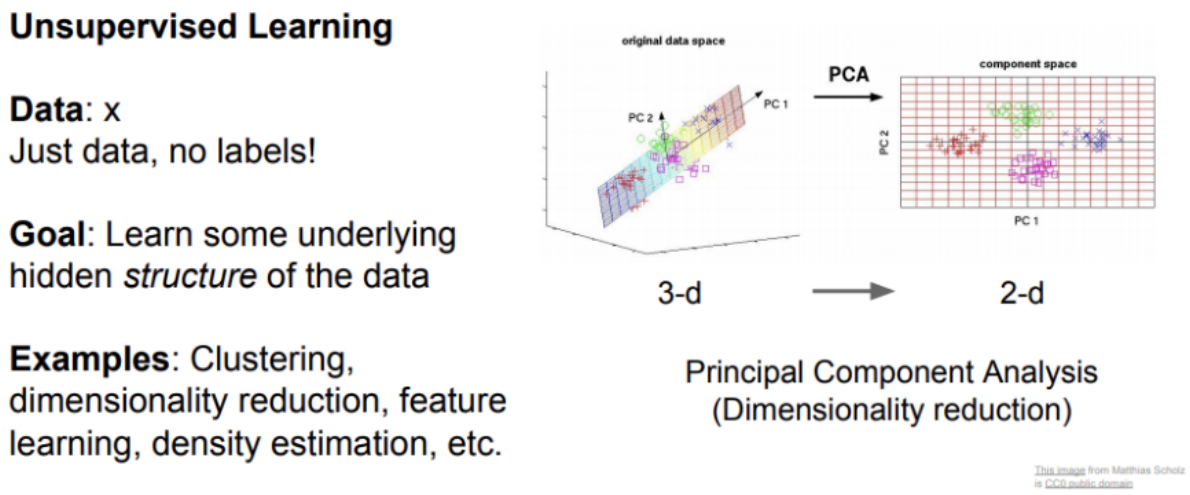
Supervised learning은 data X와 label Y가 있다. Supervised learning의 목적은 data X를 lable Y에 매핑시키는 함수를 learning하는 것이다.

ex) Classification, object detection, semantic segmantation, image captioning

Unsupervised learning은 lable 없이 학습 데이터만 가지고 데이터의 숨어있는 기본적인 구조를 학습시키는 것이다.

ex) Clustering, Dimensionality reduction, Feature learning, Density Estimation

- Dimensionality reduction



3차원의 data를 2개의 축을 찾아서 2차원으로 축소

- Autoencoder를 통한 data의 feature representation 학습

AE의 Loss는 입력 데이터를 얼마나 잘 재구성했는지를 나타내는데, 이를 이용해서 특징들을 학습시킬 수 있다. AE를 사용하면 레이블 없이도 feature representation을 학습시킬 수

있다.

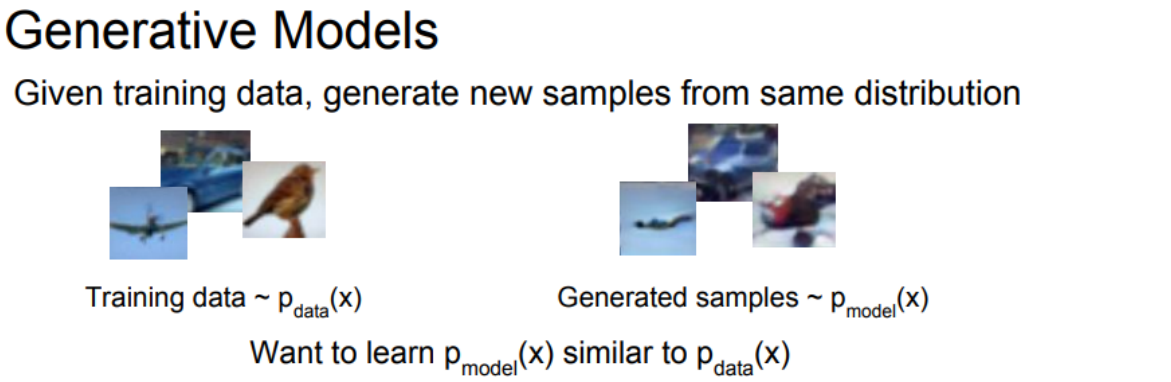
- Density estimation

이는 데이터가 가진 기본적인(underlyting) 분포를 추정하는 방법이다.

- Density Estimation

데이터가 가진 기본적인 분포를 추정하는 방법

2. Generative models



생성 모델은 비지도 학습의 일종으로, 생성 모델의 목적은 동일한 분포에서 새로운 샘플을 생성해내는것.

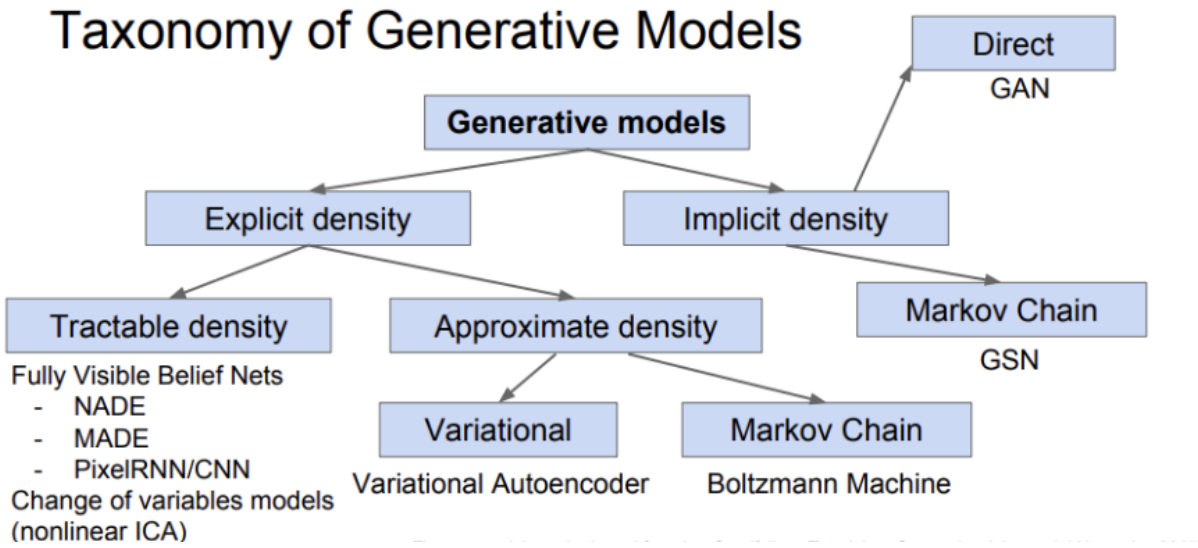
p_data로부터 나온 학습데이터가 있을 때, p_model이 p_data와 같은 데이터를 생성하도록 학습시키는 것.

⇒ Density Estimation 이용

Density Estimation 전략

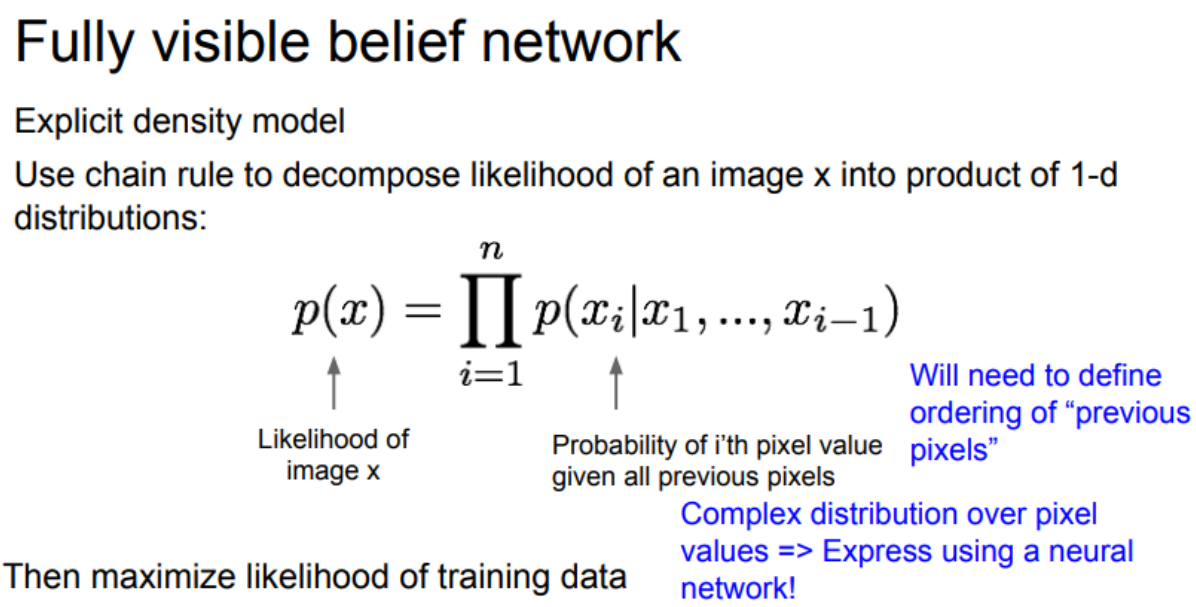
- 생성 모델 p_model의 분포가 어떨지 명시적으로 정의해 주는 방법 (explicit)
- 간접적인 방법 (Implicit)

Generative Model의 종류



다양한 생성 모델 중에서도 현재 연구가 아주 활발하게 이루어지는 pixel RNN/CNN, variational autoencoder(VAE), generative adversarial networks(GAN)

3. pixelRNN/CNN



pixel RNN/CNN 은 fully visible brief network의 일종이다.

이미지 x에 대한 liklihood인 p(x)를 모델링한다.

Chain rule로 likelihood를 나타내는 p(x)를 1차원 분포들간의 product로 decompose한다.

⇒ Pixel x_i에 대해서 각각 p(x_i | conditions) 를 정의할 수 있다.

여기서 모델을 학습시키려면 학습 데이터의 likelihood를 최대화시킨다.

- 이미지 내의 각 픽셀들의 분포 ⇒ 신경망 이용

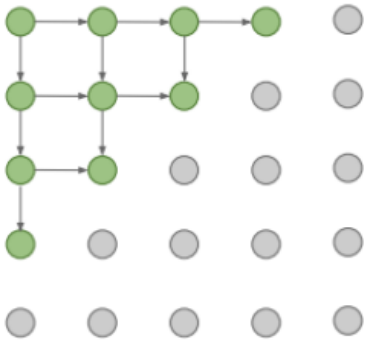
1. PixelRNN

PixelRNN *[van der Oord et al. 2016]*

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

Drawback: sequential generation is slow!



이 모델은 화살표 방향으로 순차적으로 픽셀을 생성해낸다. 이러한 방향성을 기반으로 한 픽셀들간의 종속성을 RNN 을 이용하여 모델링한다. 이러한 방식으로 대각선 아래 방향으로 계속 내려가면서 픽셀들을 생성한다.

⇒ 느림

2. Pixel CNN

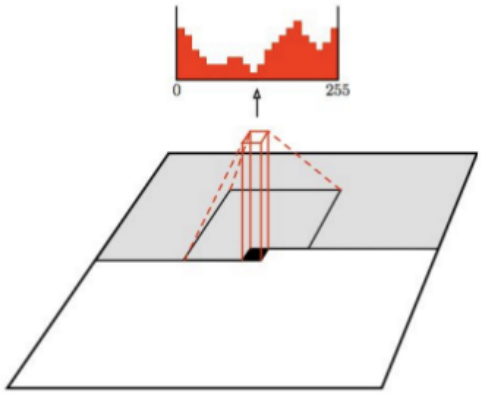
PixelCNN *[van der Oord et al. 2016]*

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



pixelCNN의 기본적인 문제는 pixelRNN과 동일하다. 왼쪽 코너에서부터 새로운 이미지를 생성한다.

pixel RNN과 달리 모든 종속성을 고려하여 모델링하는 RNN과 달리 pixel CNN은 CNN으로 모델링한다.

회색 지역은 이미 생성된 픽셀값, 이들 중 특정 영역만을 사용해서 다른 픽셀값을 생성한다.

PixelCNN에서는 각 픽셀에서 CNN을 수행한다.

픽셀을 생성하는 과정에서 각 픽셀 값은 ground truth 를 가지고 있다.

이 정답은 0-255 사이의 classification 문제를 풀기 위한 레이블이라고 볼 수 있다.

⇒ CNN은 출력 값을 가지고 softmax로 학습시킬 수 있다.

PixelRNN and PixelCNN

Pros:
- Can explicitly compute likelihood p(x)
- Explicit likelihood of training data gives good evaluation metric
- Good samples

Con:
- Sequential generation => slow

Improving PixelCNN performance
- Gated convolutional layers
- Short-cut connections
- Discretized logistic loss
- Multi-scale
- Training tricks
- Etc…

See
- Van der Oord et al. NIPS 2016
- Salimans et al. 2017 (PixelCNN++)

4. Variational AutoEncoders (VAE)

AutoEncoder

pixelCNN은 계산이 가능한 확률 모델을 기반으로 한다.

↔ VAE는 직접 계산이 불가능한 확률 모델을 정의한다.

PixelCNNs define tractable density function, optimize likelihood of training data:

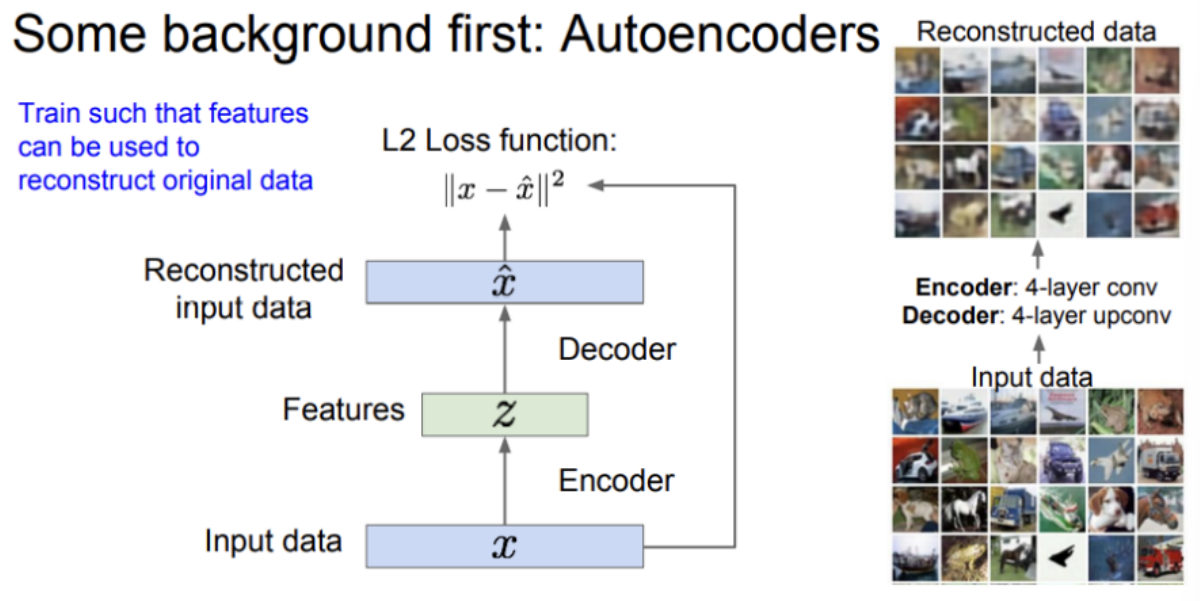
$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$

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

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

Cannot optimize directly, derive and optimize lower bound on likelihood instead

Latent variable인 z를 모델링한다. VAE에서는 data likelihood p(x)가 적분의 형태를 띄고 있다. 하지만 이 식을 직접 최적화시킬 수는 없다. 대신에 likelihood(p(x))의 lower bound를 derive해서 최적화시켜야 한다.



AE 레이블되지 않은 학습 데이터로부터 저차원의 feature representation을 학습시키기 위한 비지도 학습방법이다.

입력 데이터 x가 있고, 우리는 어떤 특징 “z” 를 학습하길 원한다.

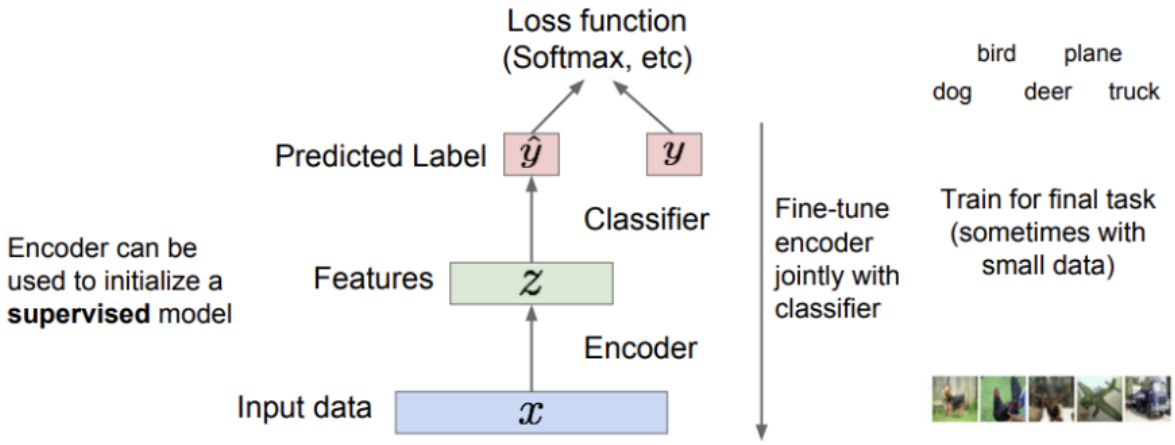
Encoder는 입력 데이터 x를 특징 z로 변환하는 매핑 함수의 역할을 한다. 일반적으로는 Neural network를 사용한다.

z는 x보다 작아서 AE를 통해 차원 축소의 효과를 기대할 수 있다.

AE는 원분을 다시 reconstruct 하는데 사용될 수 있는 특징들을 학습하는 방식을 취한다.

1. AE 과정에서 encoder는 x를 더 낮은 차원의 z로 매핑시킨다. z가 바로 encoder 네트워크의 출력이다.
2. 입력 데이터로부터 만들어진 특징 z는 두 번째 네트워크인 decoder에 사용된다. decoder의 출력은 입력 x와 동일한 차원이고, x와 유사하다. decoder는 기본적으로 encoder와 대칭적으로 동일한 구조를 지니고, CNN으로 구성된다.
3. 복원된 이미지와 원본 이미지의 차이를 계산하기 위해서 L2 같은 loss 함수를 이용한다.

Some background first: Autoencoders



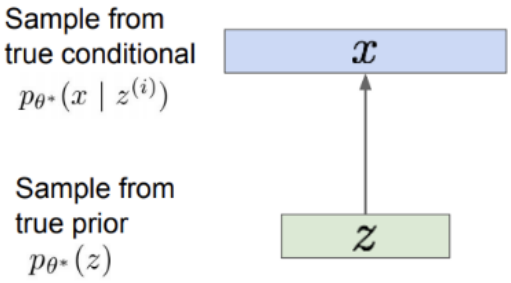
VAE

Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

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

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



VAE에서는 학습 데이터 xi(i = 1 ~ N)가 있다. 이 학습데이터가 우리가 관측할 수 없는 어떤 잠재 변수 z(latent representation)에 의해 생성된다고 가정한다. 벡터 z의 각 요소들은 데이터의 변동 요소들(some factor of variation)을 잘 포착해낸다. 즉 벡터 z가 다양한 종류의 속성을 담고 있다. 가령 얼굴을 생성할 때의 특성이라고 하면 생성된 얼굴이 얼마나 웃고 있는지, 눈썹의 위치, 머리의 방향 등의 속성들이 모두 학습될 수 있는 잠재된 요소라고 할 수 있다.

z에 대한 prior로부터 샘플링을 수행한다. 여기서 **prior** 는 확률 분포 관점에서 어떠한 event가 일어날지에 대한 기대값을 의미한다. 얼마나 웃고 있는지와 같은 속성을 담기 위해서는

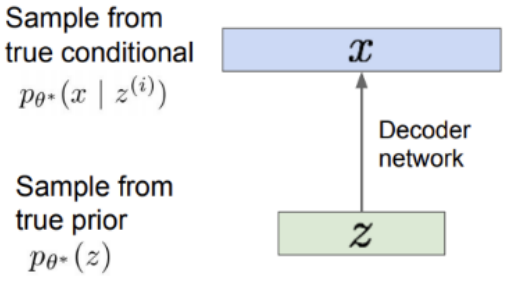
이러한 속성들이 어떤 distribution을 따르는지에 대한 prior를 정의해야 한다. 보통 prior를 선택할 때 가우시안 분포를 사용하여 표현한다.

여기에는 파라미터가 2가지 있다.

- Prior distribution
- Conditional distribution

생성 모델을 잘 만들기 위해서는 θ∗의 파라미터 값들을 잘 추정해야 한다. 그러면 이 모델을 어떻게 설계해야 할까?prior에 대해서는 gaussian과 같은 단순한 모델을 선택하지만, conditional distribution은 좀 더 복잡한 함수로 neural network로 모델링한다.

Variational Autoencoders



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

How to train the model?

Remember strategy for training generative models from FVBNs. Learn model parameters to maximize likelihood of training data

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

Now with latent z

이렇게 설계된 decoder는 어떤 잠재 변수(z)를 받아서 이미지로 디코딩하는 역할을 한다. 모델의 파라미터들을 추정하기 위해 모델을 학습시켜야 한다. 가장 쉬운 방법은 모델 파라미터가 학습 데이터의 likelihood를 최대화하도록 학습시키는 것이다. VAE의 경우에 p(x)를 모든 z에 대한 기댓값으로 나타낸다.

Variational Autoencoders: Intractability

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

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 $p_\theta(z|x)$

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

이 likelihood p(x)를 최대화시키려고 할 때 문제가 있다. p_θ(x)의 적분식을 계산할 수 없다는 것이다. p_θ(x|z)를 구하려면 모든 z에 대해 알아내야 하는데, 모든 z를 알아낼 수 없다! 여러모로 p(x)를 직접 최적화하기는 힘들다.

이 모델을 학습시키기 위한 해결책으로 decorder network(p(x|z)) 말고도 추가적인 encoder network를 정의하는 방법이 있다.

Variational Autoencoders: Generating Data!

32x32 CIFAR-10

Labeled Faces in the Wild

일반적으로 VAE가 이미지들을 잘 생성해 내기는 하지만 가장 큰 단점이 VAE로 생성한 이미지들은 원본에 비해서 블러(blurry)하다는 것이다.

Variational Autoencoders

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

Pros:
- Principled approach to generative models
- Allows inference of q(z|x), can be useful feature representation for other tasks

Cons:
- Maximizes lower bound of likelihood: okay, but not as good evaluation as PixelRNN/PixelCNN
- Samples blurrier and lower quality compared to state-of-the-art (GANs)

Active areas of research:
- More flexible approximations, e.g. richer approximate posterior instead of diagonal Gaussian
- Incorporating structure in latent variables

VAE를 요약하자면,

- VAE는 autoencoders의 확률론적 변형 버전이다.

AE는 deterministic하게 x를 받아 z를 만들고 다시 x를 복원했다면 VAE는 데이터를 생성해내기 위해서 분포와 샘플링의 개념이 추가되었다. 그리고 계산할 수 없는(intractable) 분포를 다루기 위해서 하안(lower bound)를 계산했다. "variational"은 계산한 수 없는 형태를 계산할 수 있도록 근사시키는 방법을 의미한다.(p(z given x) 를 계산못하니 q(z given x)로 근사)

- VAE와 같은 접근방식의 이점은 생성 모델에 대한 원칙적 접근(principled approach) 방법이라는 점과 모델에서 q(z given x)를 추론한다는 점이다.
- q(z given x)은 다른 테스크에서도 아주 유용한 feature representations이 될 수 있다.
- likelihood의 하안(lower bound)을 계산한다는 점 때문에 pixelRNN/CNN 같이 직접 최적화하는 방법보다는 엄밀하지 않다.
- GAN과 같은 다른 SOTA 생성모델에 비해서는 생성된 샘플이 블러하고(blurry), 퀄리티가 낮은 경향이 있다.

5. Generative Adversarial Networks (GAN)

GAN은 직접 확률분포를 모델링하지 않는다. GAN에서는 게임이론의 접근법을 취한다. GAN에서는 2-player game이라는 방식으로 학습 분포를 학습한다.

Generative Adversarial Networks

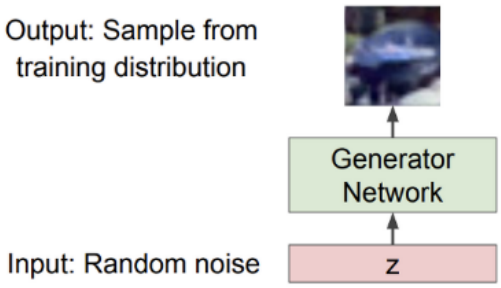
Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!

Solution: Sample from a simple distribution, e.g. random noise. Learn transformation to training distribution.

Q: What can we use to represent this complex transformation?

A: A neural network!



GAN에서 하고자 하는 것은 복잡한 고차원 학습 분포로부터 샘플링을 하는 것이다. 하지만 우리가 가진 분포가 아주 복잡하기 때문에 여기에서 직접 샘플링을 하는 것은 불가능하다.

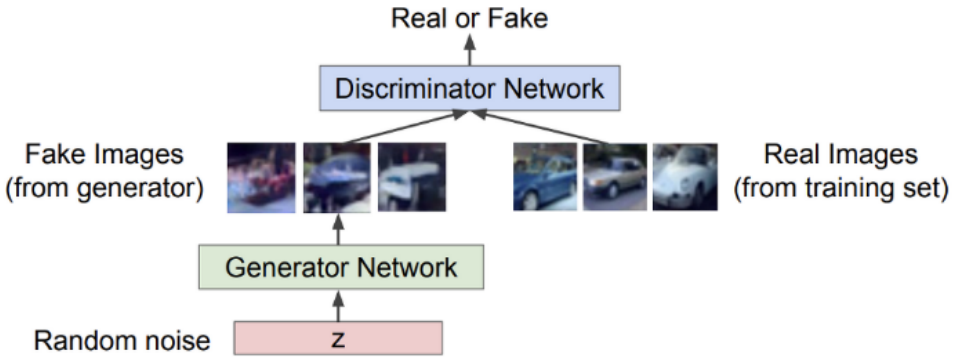
GAN에서는,gaussian random noise같은 더 단순한 분포에서는 샘플링을 할 수 있을 것이라는 아이디어를 이용한다. 그리고 단순한 분포에서 우리가 원하는 학습 분포로 변환(transformation)하는 함수를 배우고자 한다.보통 복잡한 함수나 변환를 모델링할 때 neural network를 사용한다. GAN에서는 입력으로 random noise 벡터(z)를 받는다. 벡터의 차원은 직접 명시해준다. 그리고 입력z가 생성 네트워크를 통과하면 학습 분포로부터 직접 샘플링된 값을 출력한다.

Training GANs : Two-player game

Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



GAN을 학습시키는 방법으로 two player game 방법을 이용한다.

두 명의 플레이어 generator와 discriminator가 있다.

generator는 "플레이어1" 로 참여하여 사실적인 이미지를 생성하여 discriminator를 속이는 것이 목표이다.

"플레이어2"인 discriminator 는 입력 이미지가 실제인지 거짓인지를 구별하는 것이 목표이다.

random noise가 generator의 입력으로 들어간다.

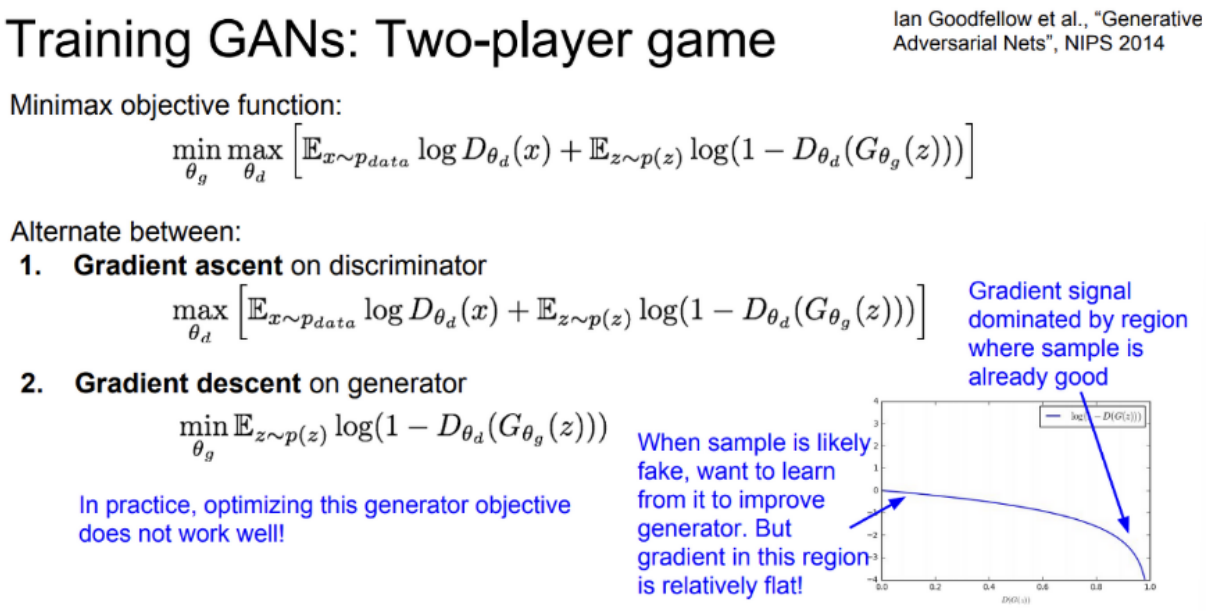
generator는 이미지를 생성해 내는데, 이 이미지는 generator가 만들어낸 "fake images"이다.

그리고 학습 데이터에서 나온 "real images"도 있다.

discriminator의 출력은 이미지가 진짜(real)인지 가짜(fake)인지이다.

GAN의 아이디어는 discriminator가 잘 학습돼서 진짜인지 가까인지를 아주 잘 구별할 수 있다면 generator는 discriminator를 속이기 위해서 더 실제같은 가짜 이미지를 만들 수 있어야 한다는 것이다. 이를 통해 성능이 아주 좋은 generative model을 만들 수 있다.

GAN 학습과정



GAN의 train시키는 과정에서 generator와 discriminator를 번갈아가면서 학습시킨다.

discriminator의 경우에는 objective function이 최대가 되는 theta를 학습하기 위해 gradient ascent를 이용한다.

generator는 반대로 gradient descent를 이용한다. gradient descent를 통해서 파라미터 theta_G를 학습시켜 object function이 최소가 되도록 한다.

하지만, 실제로는 generator의 objective function이 학습이 잘 안되는데 그 이유는 loss landscape을 살펴보면 알 수 있다. 위 그림의 오른쪽 하단의 그래프가 D(G(x))의 loss landscape이다. generator는 (1 - D(G(x)))의 값이 높을수록 좋다. 우리는 Loss가 최소가 되길 원하는데 Loss의 기울기가 오른쪽으로 갈수록 점점 커진다. 이는 discriminator가 generator를 잘 속이고 있으면 그레이언트도 점점 더 커진다는 의미이다.반면 생성된 샘플이 좋지 않을 때, 즉 generator가 아직은 잘 학습되지 않은 경우라면 discriminator가 쉽게 구분할 수 있는 상태이므로 X축 상에서 0 근처인 상태이다. 이 지점에서는 그레디언트가 상대적으로 평평하다. 이것이 의미하는 바는 그레디언트가 generator가 생성을 이미 잘 하고 있는 지역에만 몰려있다는 것이다.

하지만 우리는 샘플이 안 좋은 경우에 gradient가 크게, 즉 학습을 많이 할 수 있어야 하기 때문에 generator를 학습시키기는 상당히 어렵다. 이를 해결하기 위해 genrator에서도 gradient ascent를 이용할 것이다. discriminator가 정답을 잘 맞출 likelihood를 최소화 시키는 방법 대신에 **discriminator가 틀릴 likelihood를 최대화 시키는 쪽으로** 학습시킬 것이다.