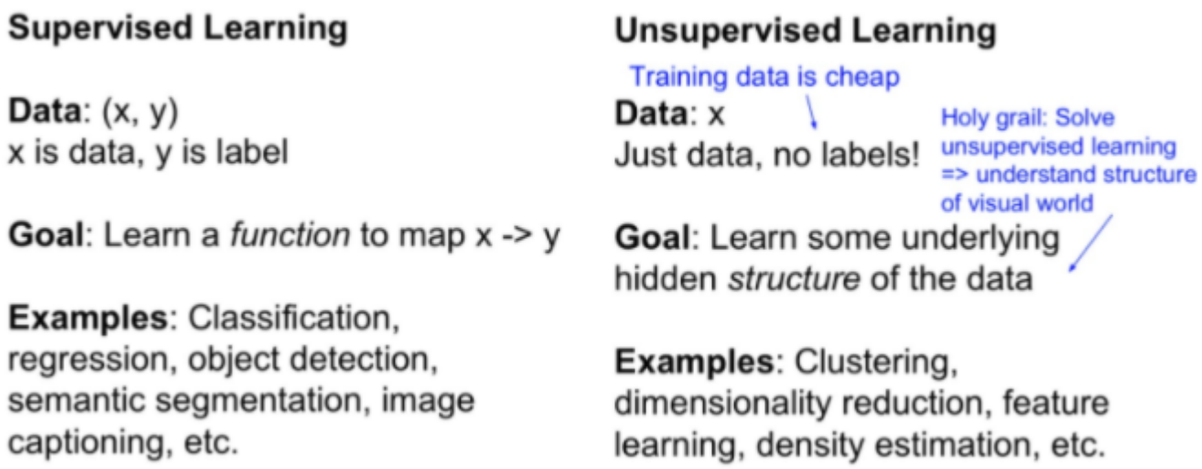
Chapter13. Generative Models

구성

Unsupervised Learning → Generative Models → PixelRNN → VAE → PixelRNN vs VAE

내용이 어렵기 때문에 구조를 이해하는 데에 집중하는 것이 중요하다 생각

1. Unsupervised Learning



기존까지 배운 내용은 `Supervised Learning` , 현 챕터는 `Unsupervised Learning`

Supervised Learning에서는 데이터와 정답이 있다.

- Image Classification에서는 Class label
- Object Detection에서는 bounding box의 1)x좌표, 2)y좌표 3)길이 4)너비.
- Semantic Segmentation에서는 각 필셀마다 설정된 카테고리
- Image Classification에는 주어진 label이 문장인 형태로 각각 정답이 존재

Unsupervised Learning에서는 정답이 없는 학습 데이터만 이용. 데이터의 숨어 있는 구조를 학습하는 것!(어디에 모여있고, 퍼져 있는지 등을 보는 것)

- Clustering : K Means, Hierarchical Clustering 등등
- 차원 축소 : PCA, FA. PCA에서 얻어낸 새 feature로 사용하는 regression 방식인 PCR은 지도학습의 범주.
- Density Estimation : 데이터의 **underlying** 분포의 추정 : seaborn의 distplot에서 kde = True에서 나오는 **Kernel Density Estimation** 등
- Feature Representation : Supervised learning에서 다양하게 배웠으며 오늘은 **Autoencoder**가 그 예시
- `Generative Models` : 오늘 주제
  - Unsupervised Learnging의 일종으로 동일한 분포에서 새로운 샘플을 생성해 내는 방법. 이를 위해서는 분포를 추정하는 것이 필요함

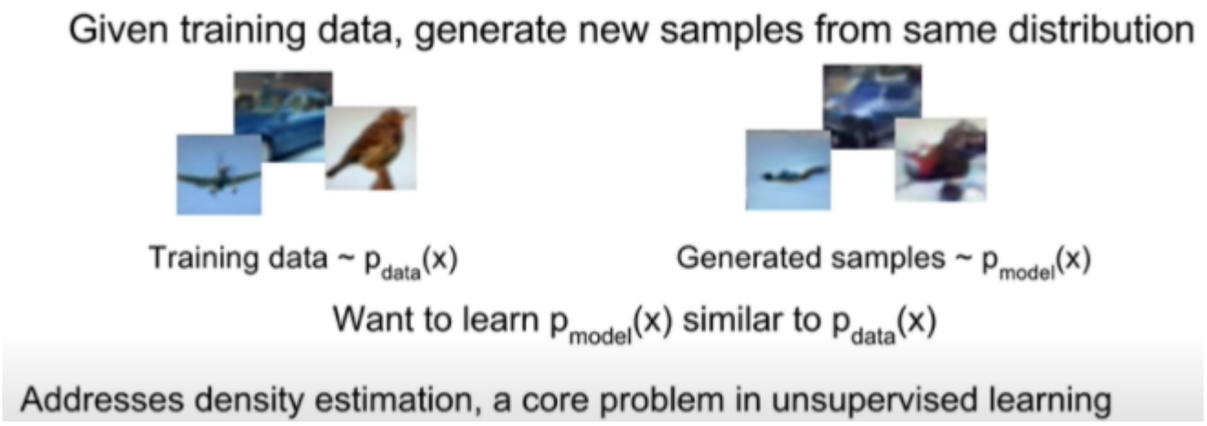
세 가지는 새로 접할 수 있겠다 생각. 서로 배반되는 개념이 아니여서 비지도 학습 종류에 관해서는 불명확한 부분이 많다.

그 외의 여러가지 방법이 존재하고 현재 활발히 연구되고 있음.

이 방법은 레이블이 없는 데이터를 이용하는 것이기 때문에 데이터를 많이 구할 수 있음.(= 데이터를 구하는 비용이 적게 듬)

keyword : `Learning a functional relationship` **vs** `learning a hidden structure`

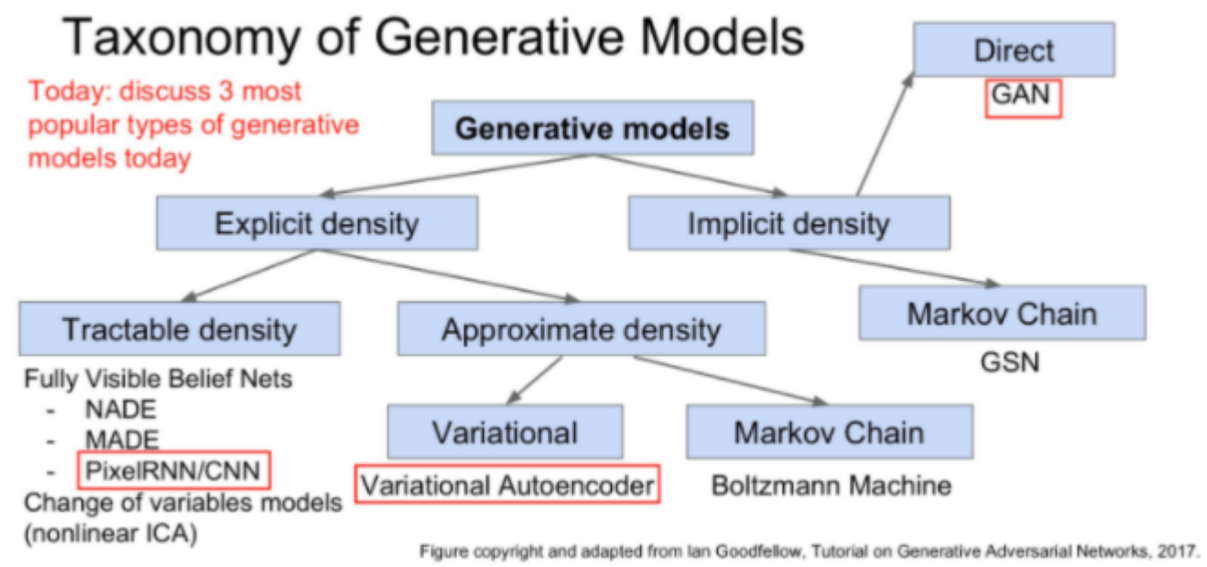
2. Generative Models



Generative Model 은 **"있을 법한, 그럴 듯한 가짜"를 생성해**내는 모델이다.

데이터가 $p(data)$분포에서 왔다고 가정하고 학습을 하여 : $p(model)$ 생성하여 다양한 sample 생성

즉 우리는 p(model)을 p(data)와 비슷하게 만들기 위한 Density Estimation이 필요 : 비지도학습의 종류를 구분하기 어렵다는 단적인 예시



Generative Model 은 Explicit Density , Implicit Density 로 나눌 수 있다.

Explicit Density 는 $P(model)$, 즉, density를 구하는 방식

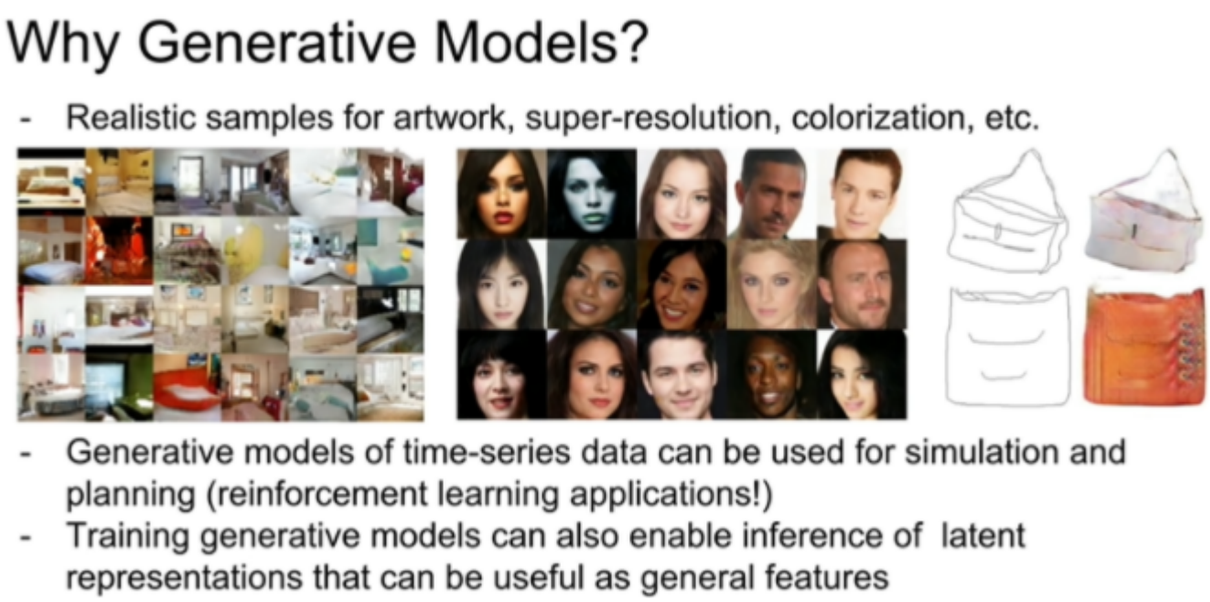
Implicit Density 는 $P(model)$을 구하는 것보다 모델을 적용하여 얻을 수 있는 점에 더욱 관심

그 안에서도 방법론적인 차이에 따라서 여러갈래로 나뉨.

13장은 Generative Model 의 가장 유명한 3가지 예시에 대해 다룬다.

오늘 다룰 내용은 Explicit Density 에 속하는 PixelRNN , Variational Autoencoder

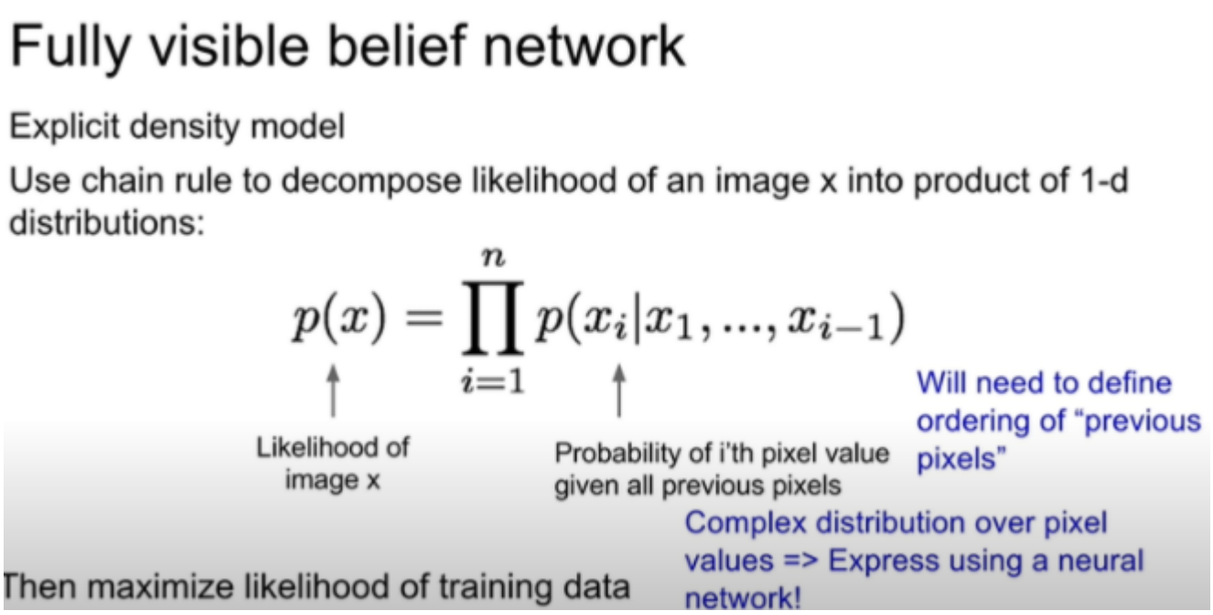
무엇을 위해 사용?



- 복제 이미지 생성
- Colorization : 흑백 사진/영상에 그럴듯한 color를 더하기
- Super-resolution : 저해상도 사진/영상의 해상도를 높이기
- 강화학습을 이용한 시뮬레이션에도 이용 -> 다음 강의에서 자세히 다루는 내용
- 플래닝을 위한 시계열 데이터 생성가능
- latent representation을 추정 가능하게 해줌

3. PixelRNN (2016)

Submitted, 25 Jan 2016, by Aaron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu



Tractable Density에 속한다 : $p(model)$을 직접 계산하는 방식 (Explicit Density)

image classification에서는 Pixel의 underlying 분포를 알고 싶다!

이미지의 제목이 Fully Visible belief network라고 나와 있는데, 확률 구조가 *tractable*, *fully visible*하니까 $p(model)$ 계산이 가능한 것이다.

사진과 같이 condition으로 이전의 모든 픽셀이 붙는다. 이 likelihood를 최대화하는 것이 목적이 될 것이다.

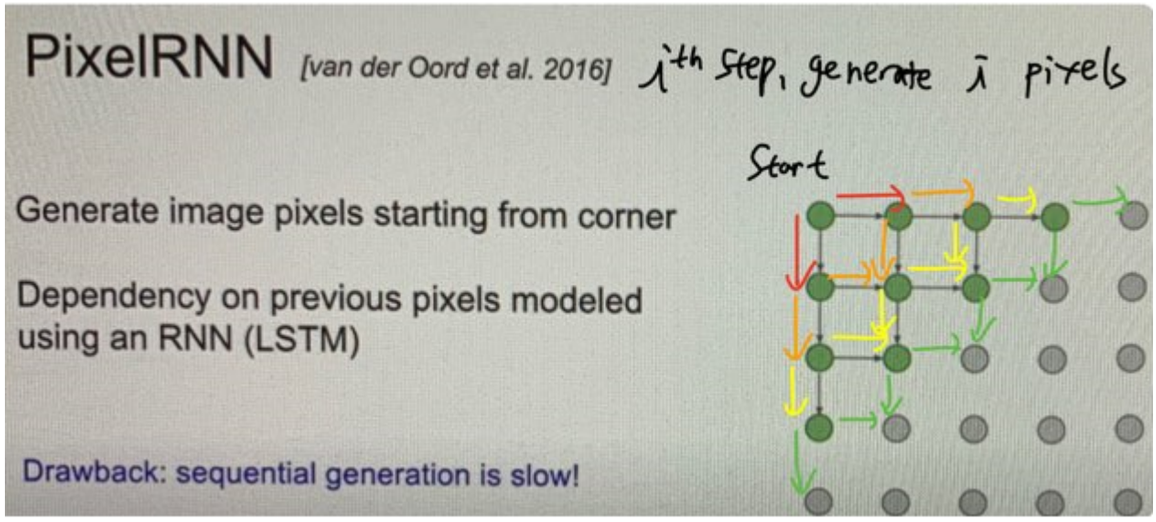
하지만 다룰 이미지는 pixel이 매우 많아 distribution이 복잡할 것이다. 따라서 복잡한 연쇄계산을 위해 Neural Network를 이용한다.

그리고 여기서 픽셀들의 순서를 어떻게 정의해서 p(x)를 계산할 것인가에 대한 생각을 해볼 수 있다.

3.1. Order of the pixels

$p(x)$의 식을 보면 $x_1, \ldots, x_{i-1}, x_i$까지의 순서구조가 중요함을 알 수 있다.

다음 이미지는 $x_1, \ldots, x_{i-1}, x_i$까지의 순서를 정하는 방식에 대한 내용이다



화살표 방향을 기본으로 종속성을 따지고 RNN(LSTM)을 이용해서 계산을 수행하게 된다. 단점은 *Sequential Generation*이며 *feed forward* 방식보다 느리다는 점이다.

3.2. PixelCNN (2016)

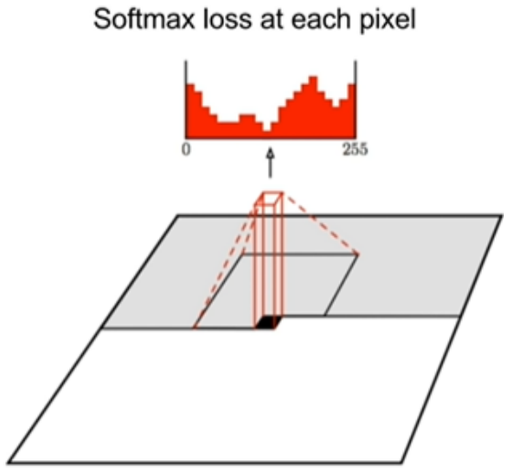
PixelCNN *[van der Oord et al. 2016]*

Still generate image pixels starting from corner

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

Training: maximize likelihood of training images

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



다른 세팅은 전부 동일하며 Pixel의 순서구조를 RNN이 아닌 CNN으로 모델링했다는 차이가 존재한다. 각 픽셀마다 주어진 값이 있기 때문에 softmax가 가능하다. (픽셀마다 0~255개의 레이블이 주어질 것이니까)그리고 그림에서 보이는 것처럼 이전의 모든 픽셀을 이용하는 것이 아니라 특정 픽셀만 사용해서 확률을 계산하게 된다. 최종적으로 이 likelihood가 최대화되도록 학습을 하는 형태이다.

그림 우상단을 보면 256개 pixel에서의 softmax loss를 줄이는 것과 likelihood maximization의 연관성이 있음을 확인해볼 수 있다.

왜 *classification label?* : 강의에서 나온 학생의 질문으로, 비지도학습인데 왜 정답 label과 같은 것이 나오냐는 것이었다. -> 여기서의 loss 계산 시 쓰이는 label : training data의 pixel값일 뿐, 지도학습 때 처럼 training data 외에 추가적으로 필요한 정답 label이 아니다. 그저 주어진 traing data만으로 계산을 하고 있는 것일 뿐이다.

PixelCNN의 장점 : `PixelRNN` **에 비해** `Training` **이 빠르다**

이유 : pixelRNN에 반해서 likelihood $p(x)$를 최대화하도록 $x_1, x_2, \ldots, x_i$가 구성되기 때문에 training pixel 구성 순서를 미리 알고 있다. (병렬화가 가능해서 연산을 빠르게 할 수 있다.)

다만, `Test(여기서는 새로운 이미지를 생성하는 작업)` 방식에 대해서는 PixelRNN에 비해 발전된 바가 없으므로 코너에서부터 순차적으로 계산이 되기 때문에 이미지 생성 시간은 느리다

** 강의 안에서 학생의 질문 : 그렇다면 처음 픽셀의 분포에 굉장히 민감해지는 것이 아닌가?

-> 맞는 말이다. 그래서 처음 픽셀의 분포를 정하는 것도 중요한 요소인데, 첫 픽셀만 학습 데이터에서 가져오거나, uniform distribution을 이용한다고 한다.

3.3. PixelRNN/CNN 정리

장점

1) Likelihood의 직접 계산 가능 (Fully visible belief network이니까)

2) Training 과정에서 $P(data)$, $P(model)$간의 evaluation metric (평가가 가능하다)

단점 : 적어도 Test에서는 느린 속도

4. VAE : AE의 확률론적 변형!

- PixelR/CNN은 *tractable density function*의 이점을 활용하여 다음의 likelihood를 최대화하였다.

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

- VAE에서는 직접 계산이 불가능한 형태(intratable)의 확률모델을 정의한다. density를 모르기 때문에 `Compound Probability Distribution` 에서 배운 바와 같이 잠재변수 z를 통해서 다음 적분식을 목표로 한다.

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

위 식은 근사 없이 참이지만 직접적으로 최대화할 수 없으며 그 대신, lower bound를 구하는 방식을 취함 : 뒤에서 좀 더 자세히 다룰 예정

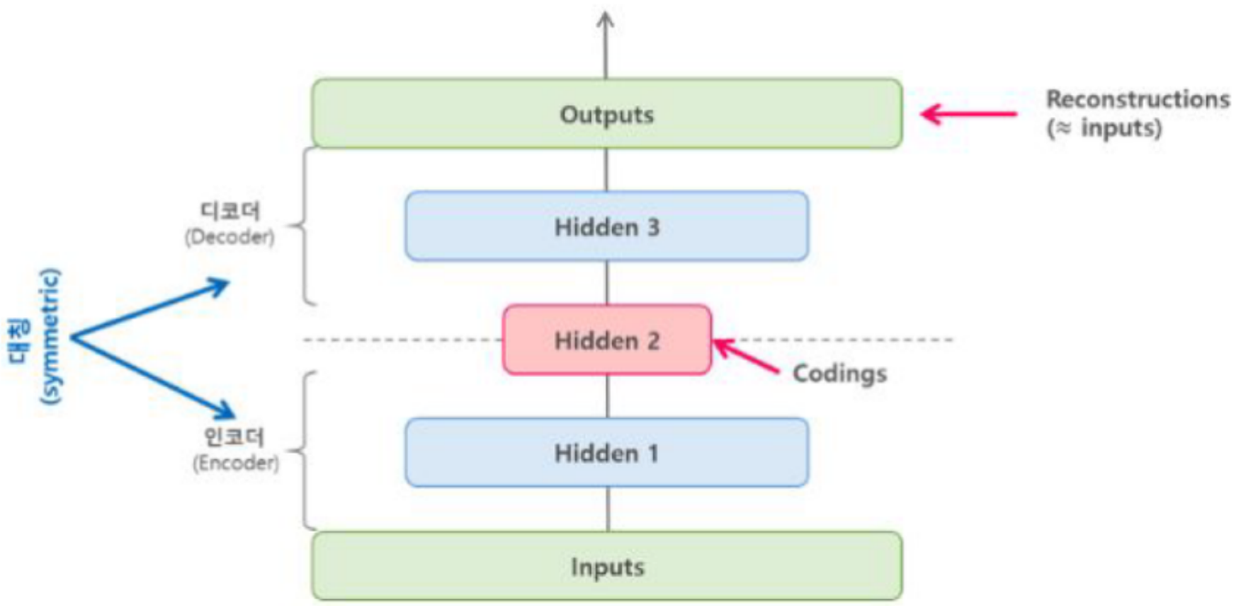
4.1 Autoencoder(AE)

Autoencoder는 Input을 학습하여 Encoder에서 `latent Feature` 를 뽑아 내고 이를 통해 Decoder에서 원본을 (근사) 복원할 수 있는 장치

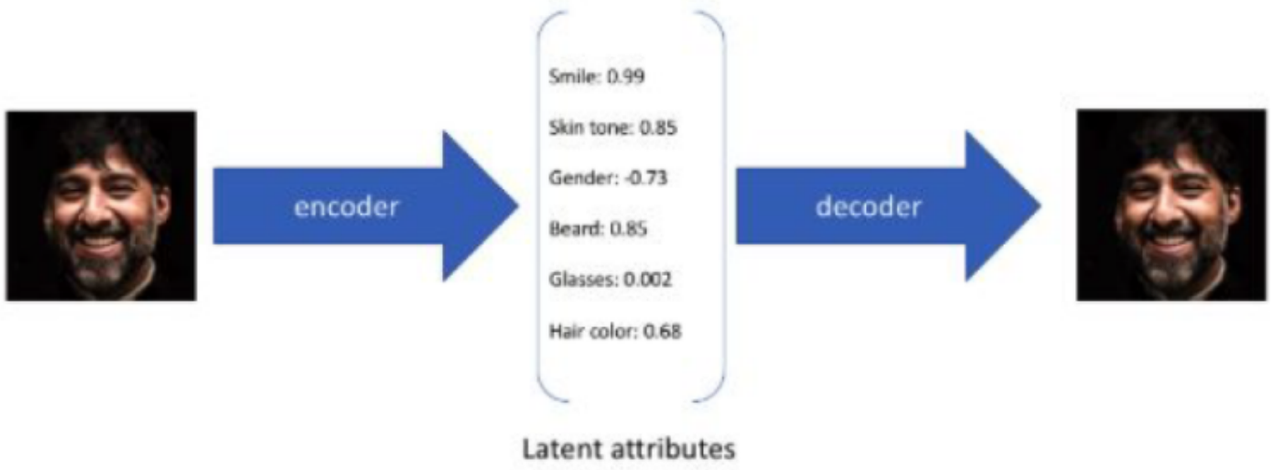
✓ Latent Variable

Latent Variable이란 직접 관찰(: 측정)이 불가능한 변수이며 input data 속에 잠재되어 있다.

사회과학) 사회적 지위(잠재변수) : 소득수준 등등



AE의 구조 [조대협의 블로그]



잠재코드의 예시 [조대협의 블로그]

특징)

1) 비지도학습에서의 Feature representation이며 `latent feature representation` 단계에서 차원축소(중요한정보만을담으니까)도 같이 이루어짐

[inputs] -> [Encoder] -> [latent vector] -> [decoder] -> [outputs]

2) $dim(Z) < dim(X) = dim(\hat{X})$,

$dim(Z)$는 연구자가 직접 명시해야 함 (x는 input data, z는 latent vector)

3) 잠재변수인 z가 Input data X의 변동을 잘 설명 : PCA와의 analogy를 생각하자

4) 보통 encoder와 decoder는 대칭의 형태로 이루어져 있다.

Details)

1) Encoder에서는 옛날에 `Sigmoid` 등의 linearity + nonlinearity, 요즘은 `ReLU` 나 `CNN`

2) $||x - \hat{x}||_2^2$: 원 이미지 pixel들과 복원 이미지 pixel들의 l_2 loss를 통해 성능 평가를 할 수 있는데, 주의할 점은 역시 class label이 없다는 것이다!

Q: 왜 Decoder가 쓰이나?

A: Encoder가 중심, Decoder는 training에서의 l_2 loss만을 계산하기 위해서만 쓰이며 훈련이 끝나면 drop.

loss를 줄이는 과정에서 z는 x를 다시 복원하기 위해서 중요한 정보들만을 점점 가지게 된다. 이는 다시 말하면 x의 중요한 정보들이 z에 잘 요약이 되어 있다는 것! 그래서 decoder이후의 파트를 버리고 z에 다른 레이어를 쌓아서 여러가지 목적으로 이용하게 된다. 잘 학습된 특징(z)는 지도학습 모델의 초기화에도 이용할 수 있다고 한다. 그렇다면 z를 이용해서 새로운 이미지를 생성해낼 수는 없을까 의문을 가지게 된다. 그렇게 VAE와 개념이 연결되게 된다.

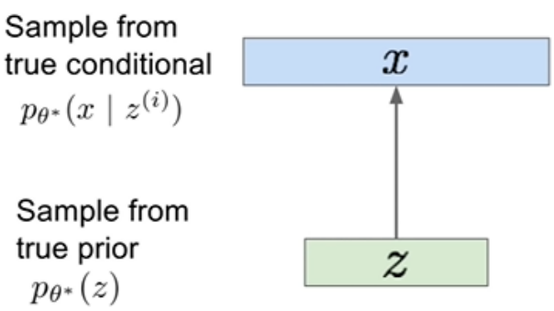
4.2 VAE 개념

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.



✓ Training data $x_1, \ldots, x_n$: 관측가능

✓ 이 Training data가 latent variable z에 의해 생성된다는 가정. 예시 : 얼굴색, 머리색, 웃는정도

✓ 1st) prior : $p_\theta * (z)$에서 z를 샘플링 sampling, $\theta*$: 참 모수

Prior $p(z)$은 **Uncorrelated Gaussian Density**와 같이 단순하게 가정

✓ 2nd) Conditional Distribution $p_\theta * (x|z_i)$를 이용해 픽셀을 만들어내서 이미지 생성

$p(x|z_i)$ 를 통해 test 단계에서 가짜 이미지들 (복잡)을 생성해야 하므로 이 분포는 매우 복잡해야 함 : 이렇게 복잡한 densitv에서는 아까 PixelCNN에서처럼 NN이 제격

헷갈리는 노테이션 정리

Prior $p_\theta * (z)$: **Uncorrelated Gaussian Density**

Encoder Network $q_\theta(z|x)$: x를 z로 encode

Posterior Z|X : Uncorrelated Gaussian Density 가정

여기서 z값을 sample!

4.3 $\theta*$ 추정 : Intractibility :(

이제 Likelihood maxization 방식으로 $\theta*$추정하자

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

$p_\theta(z)$는 우리가 정한 (예를 들면 가우시안 분포) 분포이므로 계산이 가능하다.

$p_\theta(x|z)dz$는 z를 구한 것에 따른 확률을 NN으로 계산이 가능하다. 그런데 적분 자체가 불가능하다...

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

이미지에서 먹구름은 구할 수 없다는 의미

구하고자 하는 확률을 사후분포로 바꾸어도 p(x)를 구하는 것이 쉽지 않다.

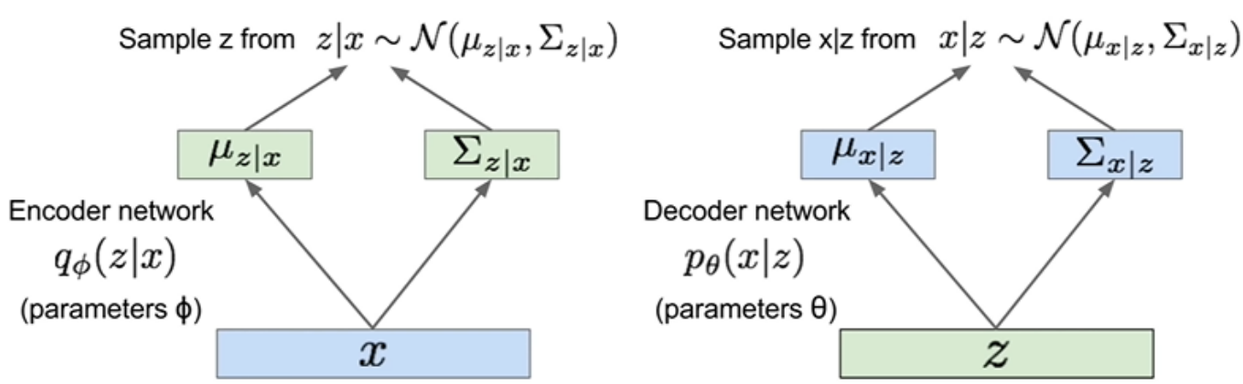
4.4 해결법 : Additional Encoder Network

추가적인 encoder $q_\phi(z|x) \approx p_\theta(z|x)$를 구한다.

이는 `likelihood` 의 `lower bound` 를 구할 수 있게 해준다 (적어도 무엇 이상).

이 Lower bound는 TRACTIBLE! (OPTIMIZATION 대상이 될 수 있음)

Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic



왼쪽 단계가 step1, 오른쪽이 step2!

4.5 이제 (log) likelihood를 구하자!

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

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

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

called Evidence Lower Bound (ELBO) Notation ($\mathcal{L}(x_i, \theta, \phi)$)

Differentiable & able to optimize

20

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

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

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

✓ 첫 줄 우변의 $\sim q_p hi(z|x_i)$가 혼란을 야기하면 그 부분을 빼고 봐도 된다. 단지 Z가 $p_\theta(x_i)$ 와 무관한 $q_\phi(z|x_i)$분포를 따른다는 뜻. 그렇게 되어 $logp_\theta(x_i)$는 Z의 관점에서는 상수가 된다 (그냥 로그에 평균을 취한 것)

✓ 4째줄에서는 log의 성질과 Expectation연산의 linearity 성질이 쓰였다.

✓ 노란색 형광펜 부분을 Evidence Lower Bound라 한다. 형광펜의 앞부분은 샘플링된 z를 통해서 얻을 수 있는 값이고, 이 부분의 값이 클수록 잘 복원이 되었다는 의미로 해석할 수 있다. 이 부분은 계산하는 테크닉이 존재한다. 그리고 두번째 부분은 q(z)를 가우시안 분포라고 했을 때, 이 분포와 q(z|x)가 비슷해야 한다고 해석할 수 있다. 이는 가우시안 간의 분포 거리를 계산하는 것으로 closed form이 존재한다. 즉 노란색 형광펜 부분은 계산이 가능하다.

✓ D_{KL} 은 KL divergence 라는 분포 간 거리 개념 (엄밀한 거리는 x. 추후설명)인데 0보다 크다. 이 부분은 정확히 계산할 수는 없지만 거리이므로 0보다 크다는 것을 이용하면, 우리가 구하고자 하는 값은 항상 노란색 형광펜 부분보다는 크다는 것을 알 수 있다. 이렇게 lower bound를 구했고 이는 최적화가 가능한 대상이므로 이제부터 이 부분을 이용할 것이다.

✓ 즉, 계속 찾고자 한 $logp_\theta(x_i)$의 lower bound를 구했다. 이는 **ELBO**이다.

4.6 $\theta*$, $\phi*$ by maximizing ELBO!

$\theta*, \phi* = argmax_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x_i, \theta, \phi)$ ($\theta*, \phi*$는 p(z|x)의 근사를 찾기 위해서 사용한 encoder와 decoder의 parameter)

Training 시, $logp_\theta(x_i)$ 대신 그 lower bound인 $\mathcal{L}(x_i, \theta, \phi)$를 최대화 하는 방식을 취한다. 말했듯이 ELBO는 **TRACTIBLE**

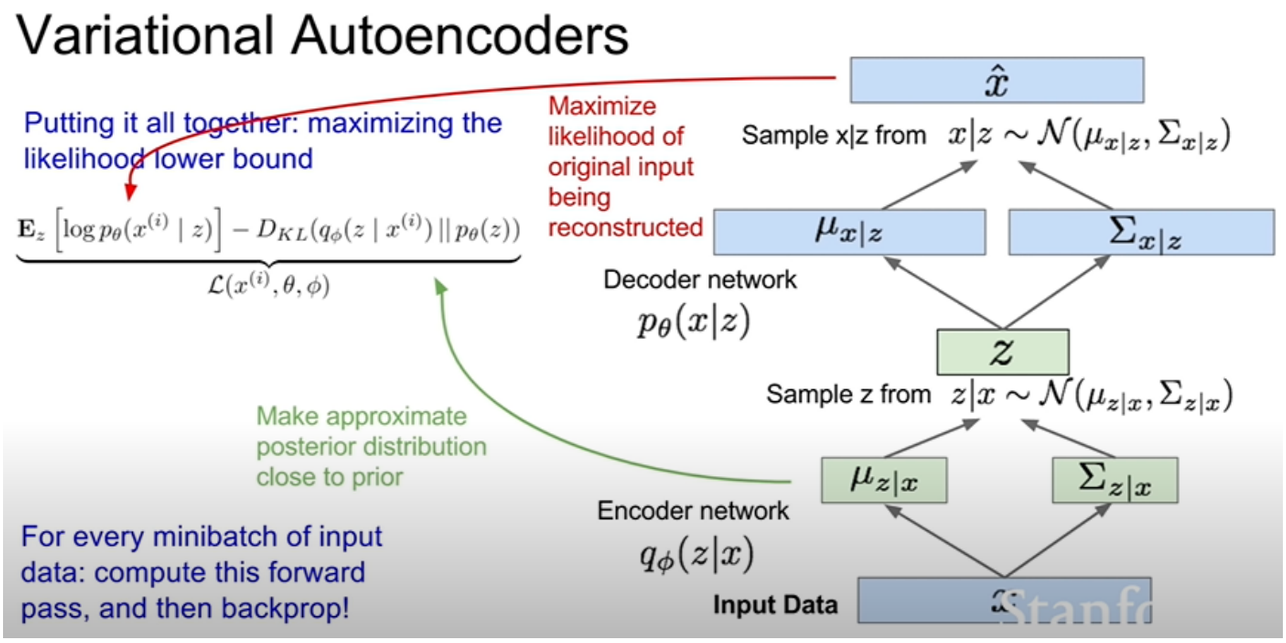
ELBO를 최대화 하려고 한다. 그러려면

1) $E_z(log(p_\theta(x_i|z))$ 를 크게 해야 한다. 즉, $p(x|z)$: likelihood가 크다는 말이며 잠재변수가 데이터를 잘 **복원**한다는 소리다.

2) $q_\phi(z|x_i)$, $p_\theta(z)$와의 KL divergence를 줄여야 한다. 즉, posterior와 정규분포 prior 분포가 유사하기를 원한다.

4.7 학습 및 Data Generating

1) 학습 과정

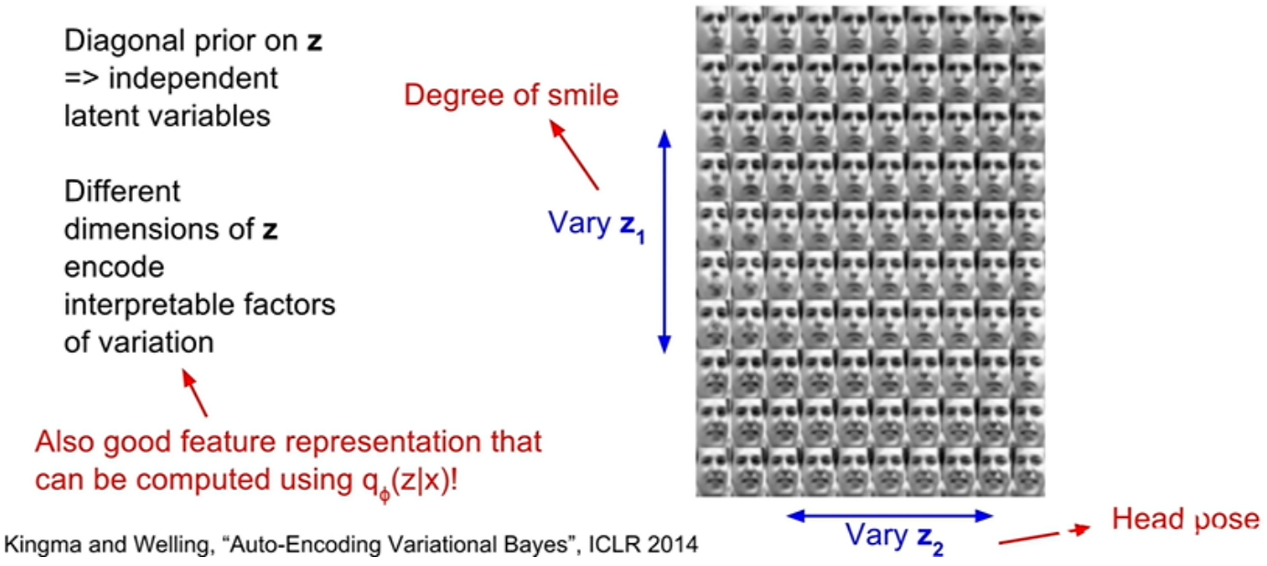


2) Data Generating 과정

Test 단계 (여기서는 Data Generating을 말한다) 에서는 **Decoder**를 쓰면 된다!

Prior에 Σ가 Diagonal인 Mutivariate Normal 분포 가정을 하였다.

MVN에서는 Uncorrelated iff Independent이므로 $z_1, z_2, \ldots, z_m$ (m = # of latent variables)는 독립



4.8 VAE 정리

✓ VAE는 AE에 probability distribution을 입혀 분포를 통해 다양한 모조 데이터를 만들 수 있게 된다

✓ Density가 `Intractible!` 이여서 ELBO라는 하한을 통해서 likelihood maximization을 하였다.

장점

$q_\phi(z|x_i)$ (posterior) 는 GM 이외의 다른 분야에서도 유용한 feature representation이 될 수 있다

단점

1) likelihood 계산에서 하한을 계산한다는 것이다. 심각한 문제가 아니다.

2) 추후 다룰 GAN에 비해 Generated data가 흐릿하다

추후에는 Implicit Density 방식에 속하는 GAN을 배운다. GAN에서는 분포 모델링 (직접계산 혹은 추정)을 하지 않으며 Sampling만을 하는 방식이다. 단지 좋은 모조 이미지를 만들어내면 그만이라는 생각

KL Divergence에 대한 보조설명

직관적인 이해를 위해서는 두 분포 간의 거리라고 이해해도 좋다. 하지만, detail을 살펴 보면

Def) Metric

A *metric* on a set X is a function $d : X \times X \to R+$ if

1. $d(x, y) \geq 0, d(x, y) = 0 \leftrightarrow x = y$: Non negativity
2. $d(x, y) = d(y, x)$: symmetry (x에서 y의 거리 = y에서 x에서의 거리)
3. $d(x, y) \leq d(x, z) + d(z, y)$: Triangle Inequality (거쳐서 가면 멀어진다)

이 Metric은 우리한테 익숙한 두 벡터 간의 거리 뿐 아니라 분포 간의 거리도 논할 수 있게 한다.

그런데 **KL Divergence**는 1번 조건이 성립하며 2,3번 조건이 성립하지 않는다!

조건 1 증명)

$KL(p||q) = -\int p(x) log \frac{q(x)}{p(x)} dx \geq -log \int p(x) \frac{q(x)}{p(x)} = 0$: $-log(\cdot)$는 $R++$에서 convex function이라는 사실을 통해 Jensen's Inequality 적용

조건 1)은 ELBO를 구할 때 쓰였다.

KL Divergence 최소화 $\leftrightarrow$ likelihood maximization

$p(x)$는 미지의 분포이며 $q(x|\theta)$ 로 근사한다고 하자. 이 때 KL Divergence

$KL(p(x)||q(x|\theta) = E_p[log \frac{p(x)}{q(x_i|\theta)}] \approx \frac{1}{N} \sum_{i=1}^{N} [logp(X_i) - log(q(x_i|\theta)]$

근사는 sample mean으로 한다. 이 때, log(p)는 θ 와 무관하므로 뒤의 -log(q)의 크기에 따라 KL divergence가 근사적으로 변하게 된다. 즉, q분포에서 θ의 likelihood를 최대화하는 것과 KL Divergence를 최소화하는 것이 같은 문제가 된다.

무엇의 KL divergence? intractible $p(x)$와 그 근사인 tractible $q(x|\theta) 간의!

citing : https://hyunw.kim/blog/2017/10/27/KL_divergence.html (https://hyunw.kim/blog/2017/10/27/KL_divergence.html)

Citing

1. https://dreamgonfly.github.io/blog/gan-explained/ (https://dreamgonfly.github.io/blog/gan-explained/) : GM 의 정의
2. https://minsuksung-ai.tistory.com/12 (https://minsuksung-ai.tistory.com/12) : Autoencoder
3. https://www.youtube.com/watch?v=5WoItGTWV54&t=370s (https://www.youtube.com/watch?v=5WoItGTWV54&t=370s) : 강의
4. https://github.com/visionNoob/CS231N_17_KOR_SUB/blob/master/kor/Lecture%2013%20%20%20Generativ (https://github.com/visionNoob/CS231N_17_KOR_SUB/blob/master/kor/Lecture%2013%20%20%20Generati : 강의 번역
5. https://hyunw.kim/blog/2017/10/27/KL_divergence.html (https://hyunw.kim/blog/2017/10/27/KL_divergence.html) : KL Divergence