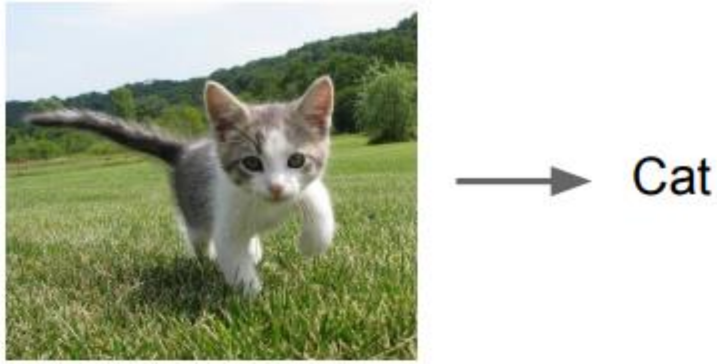
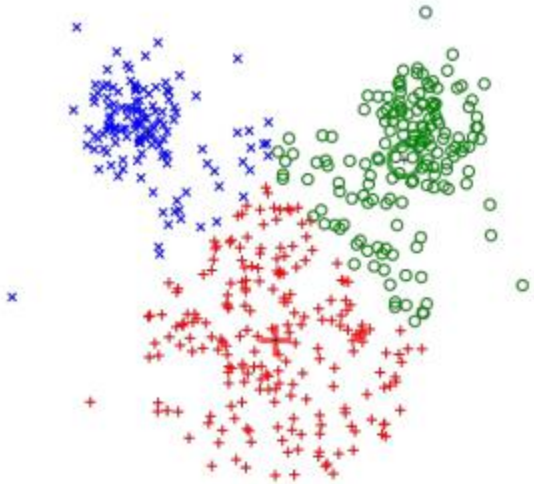
CS231n 13강
Generative Models

Supervised Learning vs Unsupervised Learning



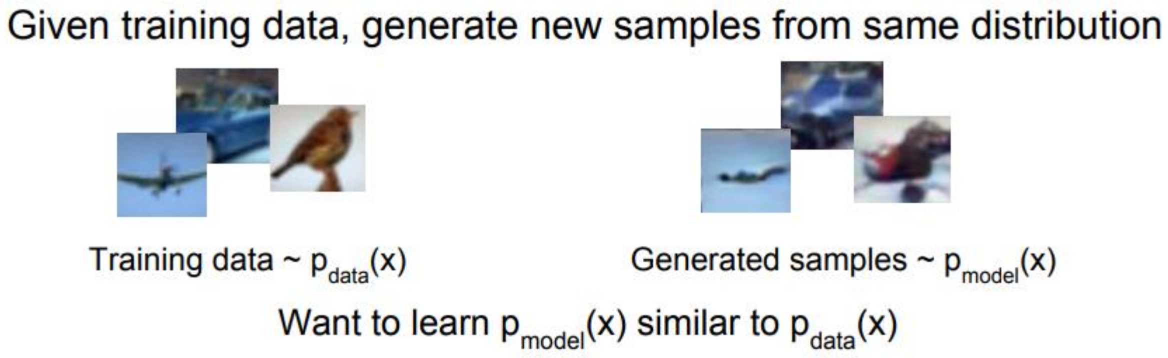
Classification

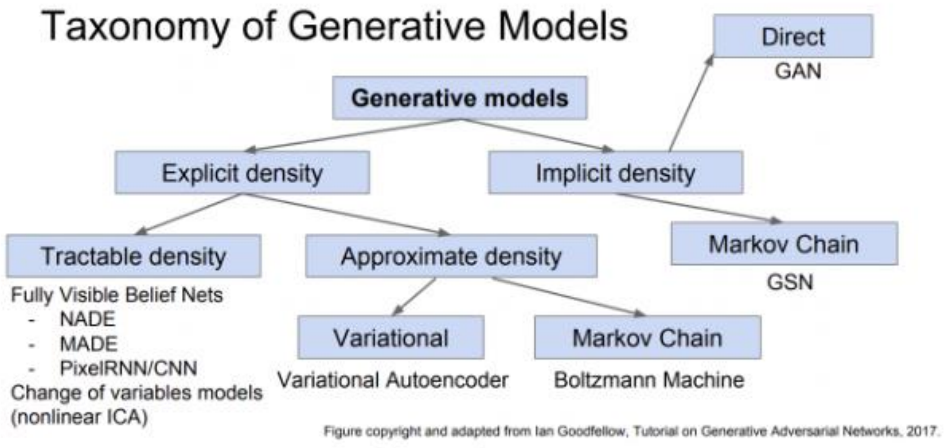


- 학습데이터 label, 정답이 주어짐
- Function을 학습
- Object detection, Semantic Segmentation,Image Captioning

- 학습데이터 label, 정답이 X!!
- Label이 없어서 data의 hidden structure을 찾아내야함
- Clustering,dimensionality reduction,feature learning,density estimation

Generative Models





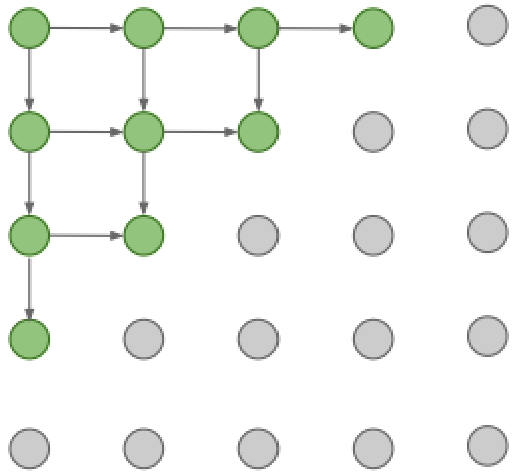
- Reference training data가 주어지면 비슷한 분포를 생성하는것이 목표
- **Explicit density**: P(x)이 어떤 분포 띄는지 정의하고 찾음,training data의 likelihood
- **Implicit density**: P(x)가 sample을 생성할수 있는 수준

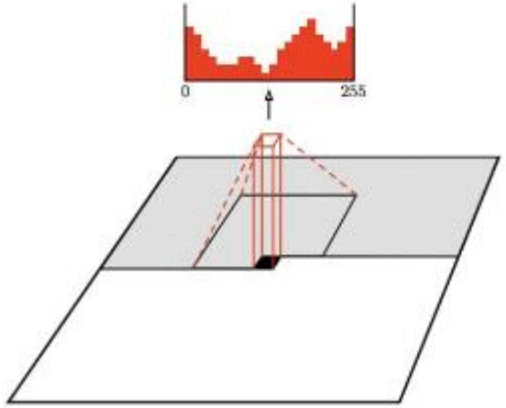
$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1})$$

Likelihood of image x

Probability of i'th pixel value given all previous pixels

Pixel RNN VS Pixel CNN





- 왼쪽 위 코너 시작점부터 상하좌우로 뻗어나감
- 이전 결과에 영향을 받아 이전 pixel에 대한 dependency ->RNN

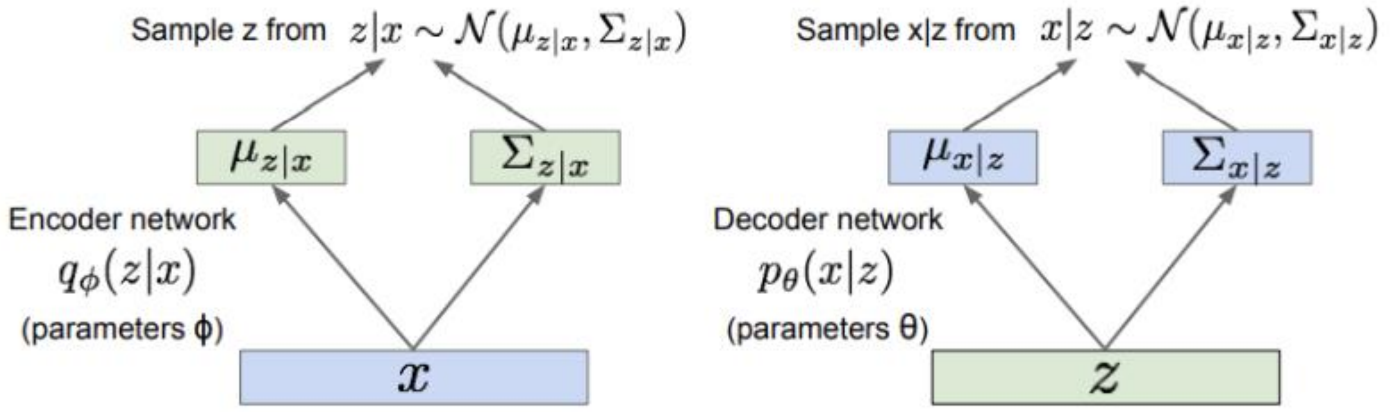
- Piexel RNN에서 RNN대신 CNN
- 인접한 좌표를 한번에 CNN하는 방식
- Pixel RNN보다 빠름

VAE(Variational Autoencoders) :function의 하한선 maximize

- Prior: 확률 분포 관점에서 어떠한 event가 일어날 지에 대한 기대값
- Prior distribution,Conditional distribution
- P(x|z)가 계산이 불가능해 근사한 추가적인 encoder network

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

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



Encoder and decoder networks also called
"recognition"/"inference" and "generation" networks

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

VAE(Variational Autoencoders) Training

$$\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z\mid x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)}\mid z)p_\theta(z)}{p_\theta(z\mid x^{(i)})}\right] \quad (\text{Bayes' Rule})\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)}\mid z)p_\theta(z)}{p_\theta(z\mid x^{(i)})}\frac{q_\phi(z\mid x^{(i)})}{q_\phi(z\mid x^{(i)})}\right] \quad (\text{Multiply by constant})\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)}\mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z\mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z\mid x^{(i)})}{p_\theta(z\mid x^{(i)})}\right] \quad (\text{Logarithms})\\
&= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)}\mid z)\right] - D_{KL}(q_\phi(z\mid x^{(i)})\,||\,p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z\mid x^{(i)})\,||\,p_\theta(z\mid x^{(i)}))}_{\geq 0}
\end{aligned}$$

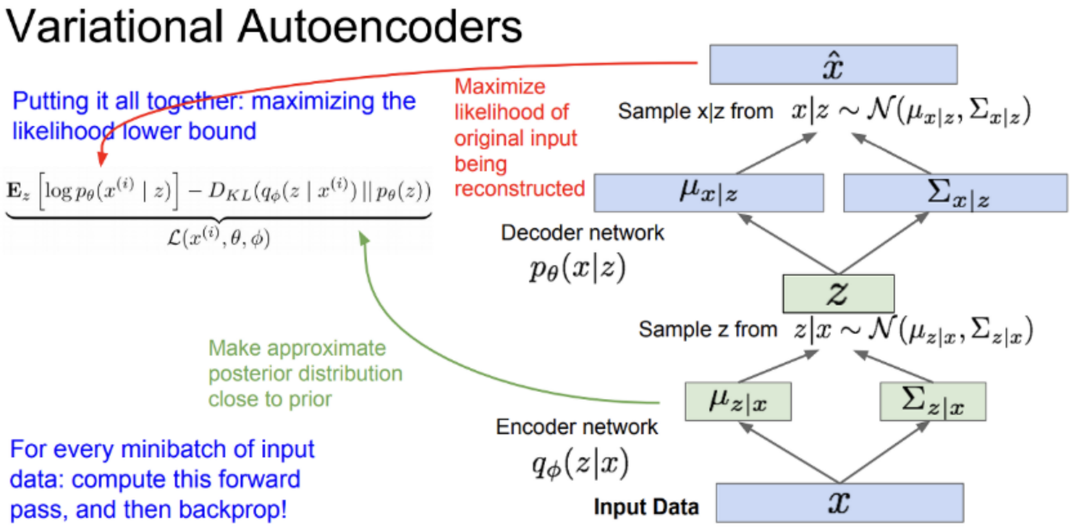
$$\boxed{\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)},\theta,\phi)}$$

Variational lower bound ("ELBO")

$$\boxed{\theta^*,\phi^* = \arg\max_{\theta,\phi}\sum_{i=1}^{N}\mathcal{L}(x^{(i)},\theta,\phi)}$$

Training: Maximize lower bound

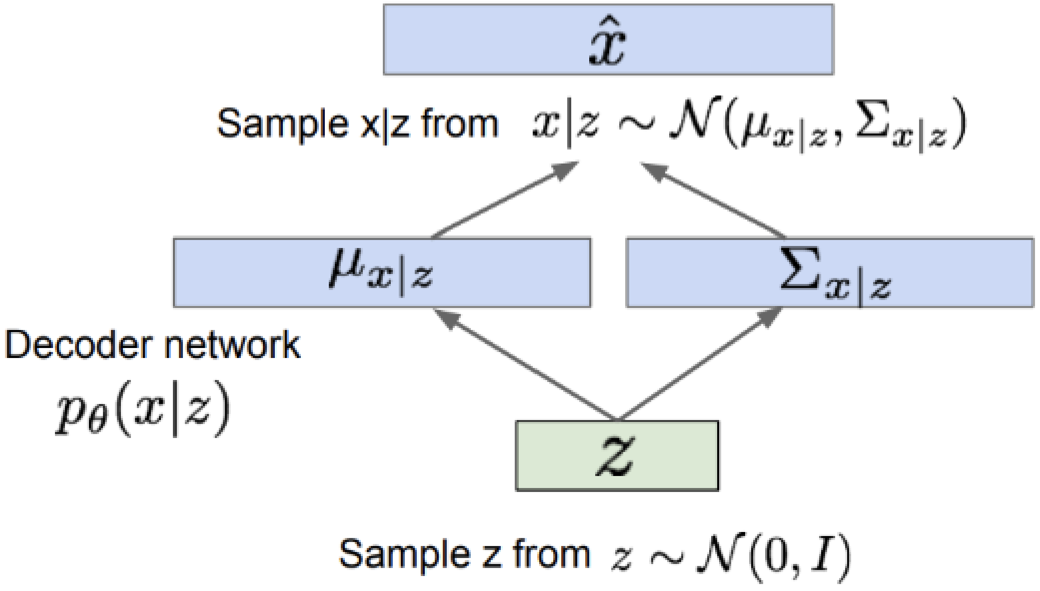
- Lower bound 최대화



- 입력데이터 x를 encoder에 통과시켜 q(z|x)얻음
- 잠재변수 z 샘플링
- Z decode에 통과
- Training time log p
- gradient 계산하여 backprop

VAE(Variational Autoencoders) Training

Use decoder network. Now sample z from prior!



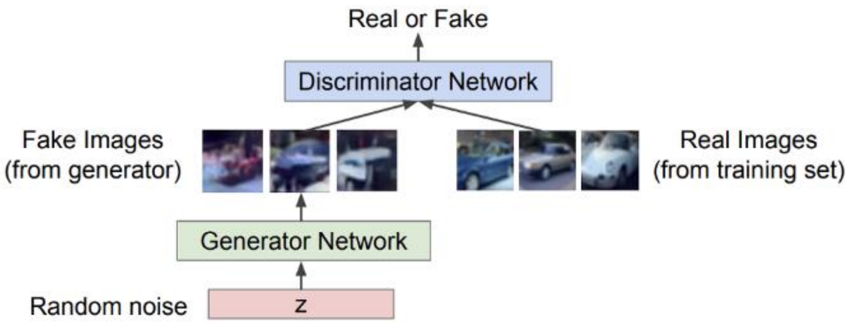
Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

- 데이터 생성 시 decoder network만 필요
- Prior에서 샘플링

GAN(Generative Adversarial Networks)

- 단순한 분포에서 학습 분포로 변환하는 함수를 배우고자 함
- Training에서 two-player game이용, generator & discriminator
- Minimax game형태로 같이 학습시킴
- generator 는 gradient descent, discriminator은 gradient ascent

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



Training GANs: Two-player game

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

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

In practice, optimizing this generator objective does not work well!

Gradient signal dominated by region where sample is already good

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!