Generative Adversarial Nets

Xinjie Fan

Department of Statistics and Data Sciences

The University of Texas at Austin

December 4, 2017

Overview

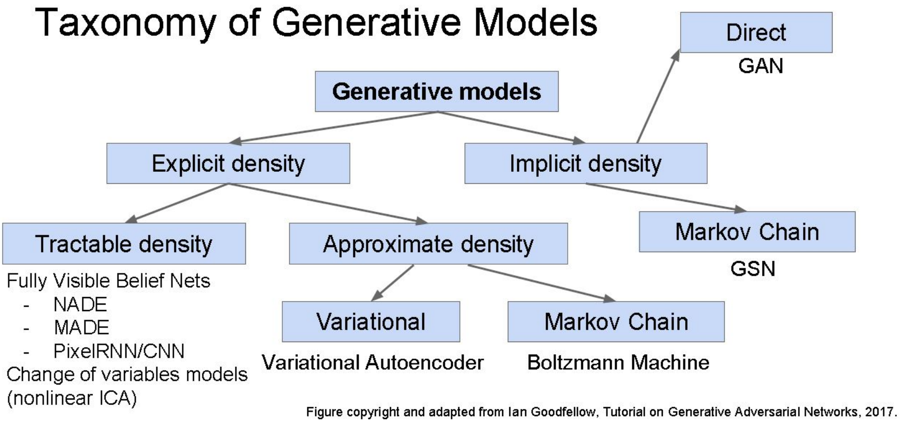
Generative models

Here we focus on using generative model to solve the unsupervised problem:
given training data(without labels), generate new samples from same distribution.

Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

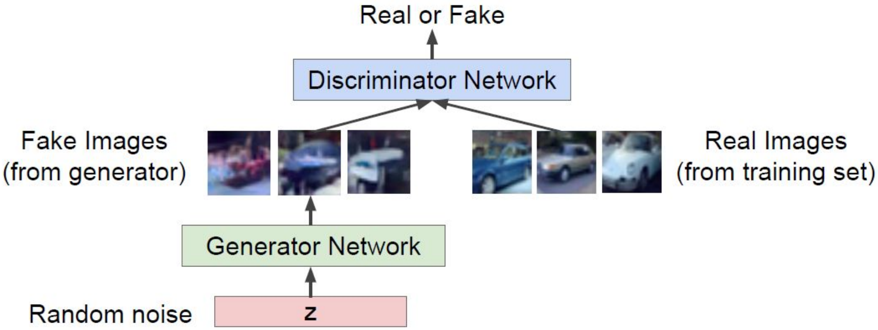
Question: why generative models?



GANs: don't work with explicit density function.

Basic idea: two-player game

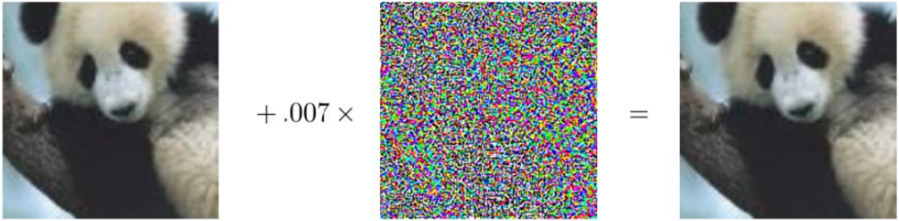
Generator network: try to fool the discriminator by generating real-looking images.
Discriminator network: try to distinguish between real and fake images.



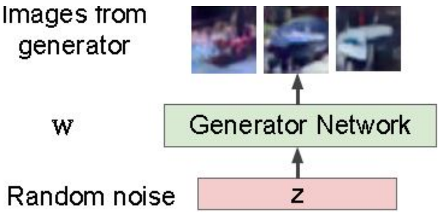
First perspective on the motivation: adversarial example

Adversarial example

Panda or gibbon?



Second perspective: what is maximum likelihood?



In this case likelihood is intractable, because we cannot go back from images to latent variable. Therefore, we change the direction.

Vanilla GAN

Objective function:

$$\min_G \max_D [E_{x \sim p_{data}} \log D(x) + E_{z \sim noise} \log(1 - D(G(z)))]$$

Here, discriminator outputs likelihood in $(0, 1)$ of input images.
Alternating optimization leads to the minimization problem unbounded.
Therefore, usually we alternate between:

$$\max_D [E_{x \sim p_{data}} \log D(x) + E_{z \sim noise} \log(1 - D(G(z)))]$$

$$\max_G E_{z \sim noise} \log(D(G(z)))$$

The algorithm

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

Demonstrations

Demo on MNIST example (compare with VAE): ▸ MNIST

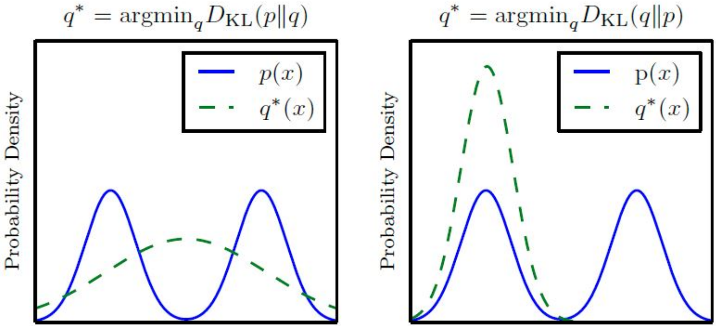
Demo on 1D example: ▸ 1D

Why it works?

Does the choice of divergence matter?
Loosely speaking, GAN would find a p_{model} such that the following Jensen-Shannon divergence is minimized:

$$JSD(p_{data}||p_{model}) = \frac{KL(p_{data}||\frac{p_{data}+p_{model}}{2}) + KL(p_{model}||\frac{p_{data}+p_{model}}{2})}{2}.$$

In contrast, VAE minimizes KL divergence.



Unfortunately...

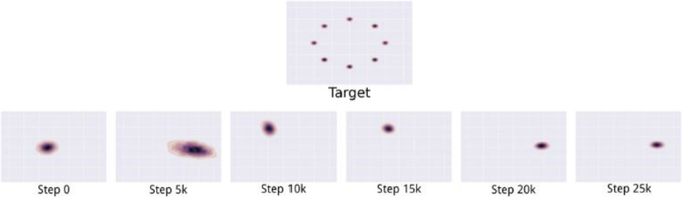
But that is not the answer!
We can minimize KL divergence in GAN framework by minimizing

$$J^G = \frac{1}{2} E_{z \sim p_{model}} \exp(\sigma^{-1}(D(G(z)))).$$

And it still works.

Why it does not work?

The problem of mode collapse: GANs often choose to generate from very few modes; fewer than the limitation imposed by the model capacity.



One explanation: it is problematic to use alternating minimization as minimax is not equivalent to maxmin in this case.

Variations of GANs

GAN	Discriminator Loss	Generator Loss
MM GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] + \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{GAN}} = -\mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}}$
NS GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{NSGAN}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}}$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{NSGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} = -\mathbb{E}_{x\sim p_d}[D(x)] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{WGAN}} - = \mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}}$
WGAN GP	$\mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_g}[(\lvert\lvert\nabla D(\alpha x + (1 - \alpha\hat{x})\rvert\rvert_2 - 1)^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{WGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$
LS GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{LSGAN}} = -\mathbb{E}_{x\sim p_d}[(D(x) - 1)^2] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{LSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[(D(\hat{x} - 1)^2]$
DRAGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{DRAGAN}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_d + \mathcal{N}(0,c)}[(\lvert\lvert\nabla D(\hat{x})\rvert\rvert_2 - 1)^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{DRAGAN}} = -\mathcal{L}_{\mathrm{D}}^{\mathrm{NS\ GAN}}$
BEGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{BEGAN}} = \mathbb{E}_{x\sim p_d}[\lvert\lvert x - \mathrm{AE}(x)\rvert\rvert_1] - k_t\mathbb{E}_{\hat{x}\sim p_g}[\lvert\lvert\hat{x} - \mathrm{AE}(\hat{x})\rvert\rvert_1]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{BEGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\lvert\lvert\hat{x} - \mathrm{AE}(\hat{x})\rvert\rvert_1]$

What? They are equal? ▸ Are GANs Created Equal? A Large-Scale Study

Tricks and tips

- Tanh as the last layer of the generator output
- Dropout
- Conditional on labels (reduce the number of modes)
- Batch normalization
- Deeper discriminator
- Label smoothing for discriminator

Improved techniques for training GANs. Tips and tricks to make GANs work

Fun stuff and references

Cycle GAN: ▸ Cycle GAN ▸ Face change with Cycle GAN

Adversarial examples: ▸ Adversarial examples and adversarial training

Tutorial: ▸ Tutorial on GAN

Stanford course video: ▸ Generative models

The End