(Deep) Generative Models for Unsupervised Learning (Part 4 – GANs)

CS698X: Topics in Probabilistic Modeling and Inference

Piyush Rai

Generative Adversarial Network (GAN)

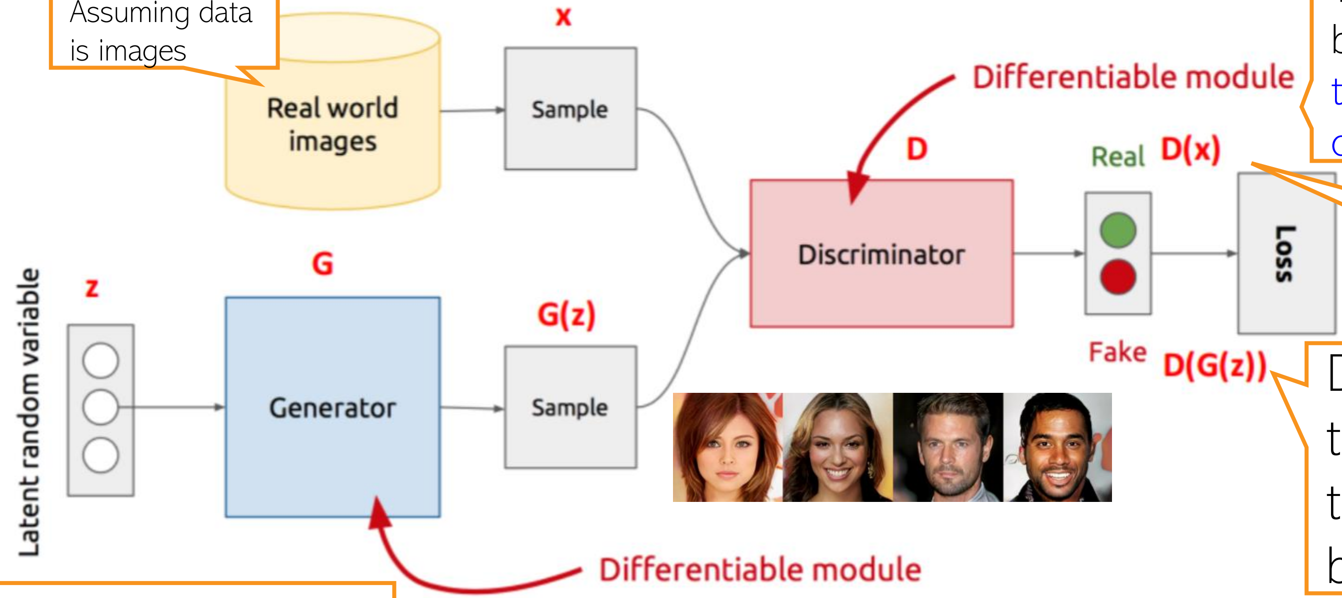
- GAN is an implicit generative latent variable model
- Can generate from it but can't compute $p(x)$ - the model doesn't define it explicitly
- GAN is training using an adversarial way (Goodfellow et al, 2013)

Unlike VAE, no explicit parametric likelihood model $p(x|z)$

Thus can't train using methods that require likelihood (MLE, VI, etc)

The discriminator can be a binary classifier or any method that can compare b/w two distributions (real and fake here)

Assuming data is images



Discriminator network is trained to make it close to 1

Discriminator network is trained to make it close to 0 and generator network is trained to make it to be close to 1 to fool the discriminator into believing that $G(z)$ is a real sample

Min-max optimization

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

Generative Adversarial Network (GAN)

- The GAN training criterion was

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

- With G fixed, the optimal D (exercise)

Distribution of real data

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

Distribution of synthetic data

- Given the optimal D, The optimal generator G is found by minimizing

$$V(D_G^*, G) = \mathbb{E}_{x\sim p_{data}}\left[\log \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right] + \mathbb{E}_{x\sim p_g}\left[\log \frac{p_g(x)}{p_{data}(x)+p_g(x)}\right]$$

$$= \mathrm{KL}\left[p_{data}(x) \middle\| \frac{p_{data}(x)+p_g(x)}{2}\right] + \mathrm{KL}\left[p_g(x) \middle\| \frac{p_{data}(x)+p_g(x)}{2}\right] - \log 4$$

Jensen-Shannon divergence between p_{data} and p_g, minimized when $p_g = p_{data}$

Thus GAN can learn the true data distribution if the generator and discriminator have enough modeling power

GAN Optimization

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}))]$$

- The GAN training procedure can be summarized as

$\boldsymbol{\theta}_g$ and $\boldsymbol{\theta}_d$ denote the params of the deep neural nets defining the generator and discriminator, respectively

Initialize $\boldsymbol{\theta}_g$, $\boldsymbol{\theta}_d$;
for *each training iteration* **do**
  for K *steps* **do**
    Sample minibatch of M noise vectors $\mathbf{z}_m \sim q_z(\mathbf{z})$;
    Sample minibatch of M examples $\mathbf{x}_m \sim p_D$;
    Update the discriminator by performing stochastic gradient *ascent* using this gradient:
    $\nabla_{\boldsymbol{\theta}_d} \frac{1}{M} \sum_{m=1}^{M} [\log D(\mathbf{x}_m) + \log(1 - D(G(\mathbf{z}_m)))].$;
  Sample minibatch of M noise vectors $\mathbf{z}_m \sim q_z(\mathbf{z})$;
  Update the generator by performing stochastic gradient *descent* using this gradient:
  $\nabla_{\boldsymbol{\theta}_g} \frac{1}{M} \sum_{m=1}^{M} \log(1 - D(G(\mathbf{z}_m))).$;
Return $\boldsymbol{\theta}_g$, $\boldsymbol{\theta}_d$

In practice, for stable training, we run $K > 1$ steps of optimizing w.r.t. D and 1 step of optimizing w.r.t. G

In practice, in this step, instead of minimizing $\log(1 - D(G(z))$, we maximize $\log\left(D(G(z))\right)$

Reason: Generator is bad initially so discriminator will always predict correctly initially and $\log(1 - D(G(z))$ will saturate

Least-Squares GAN (LSGAN)

- Instead of a log-loss objective, another alternative is Least Squares GAN (LSGAN)
  - Discriminator minimizes the following loss

Function of discriminator network params

m real examples and m fake examples from the generator

$$J^D = \frac{1}{2m}\sum_{i=1}^{m}\left[(D(\mathbf{x}_i)-1)^2\right] + \frac{1}{2m}\sum_{i=1}^{m}\left[(D(G(\mathbf{z}_i)))^2\right]$$

  - Generator minimizes the following loss

Function of generator network params

$$J^G = \frac{1}{m}\sum_{i=1}^{m}\left[(D(G(\mathbf{z}_i))-1)^2\right]$$

  - We can perform ALT-OPT for learning the parameters of D and G

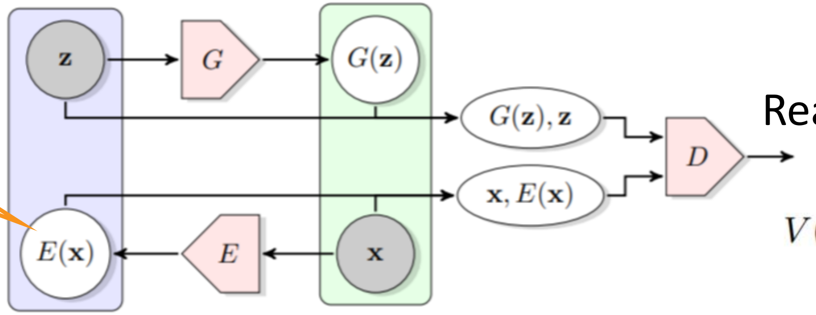
- LSGAN address some of the issues, such as saturating gradients

GANs that also learn latent representations

- The standard GAN can only generate data. Can't learn the latent $\boldsymbol{z}$ from $\boldsymbol{x}$
- Bidirectional GAN* (BiGAN) is a GAN variant that allows this

Consists of an encoder as well

Can be shown* to "invert" $\boldsymbol{G}$



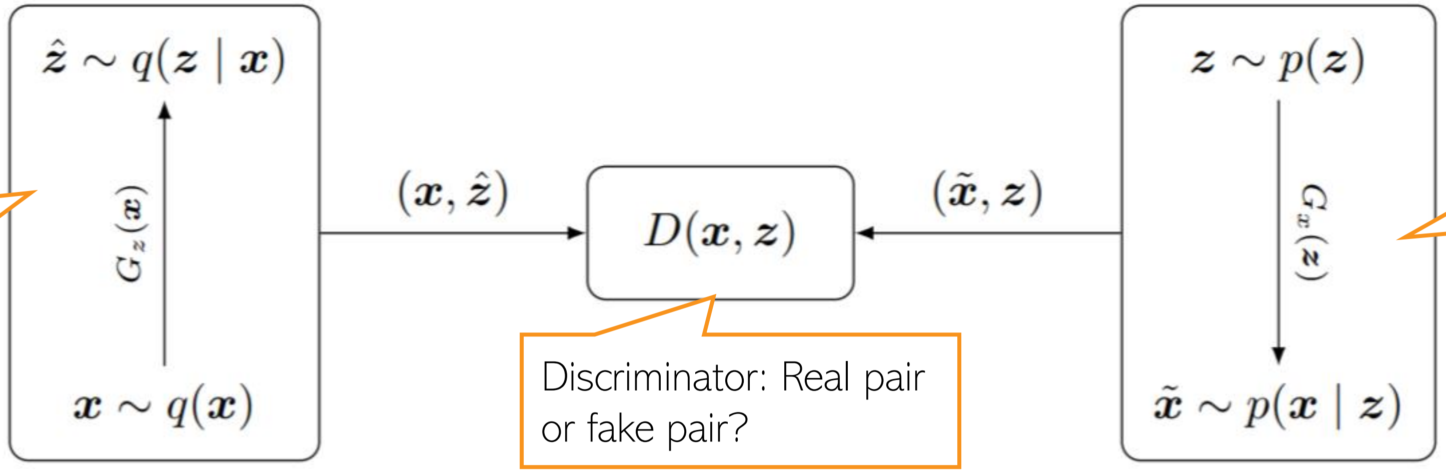
Real pair/fake pair?

$$\min_{G,E} \max_{D} V(D,E,G)$$

$$V(D,E,G) := \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{X}}}\Big[\underbrace{\mathbb{E}_{\mathbf{z}\sim p_E(\cdot|\mathbf{x})}\left[\log D(\mathbf{x},\mathbf{z})\right]}_{\log D(\mathbf{x},E(\mathbf{x}))}\Big] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{Z}}}\Big[\underbrace{\mathbb{E}_{\mathbf{x}\sim p_G(\cdot|\mathbf{z})}\left[\log\left(1-D(\mathbf{x},\mathbf{z})\right)\right]}_{\log(1-D(G(\mathbf{z}),\mathbf{z}))}\Big]$$

- Adversarially Learned Inference# (ALI) is another variant that can learn representations

Encoder with joint $q(x,z) = q(x)q(z|x)$



Decoder/generator with joint $p(x,z) = p(z)p(x|z)$

Discriminator: Real pair or fake pair?

*Adversarial Feature Learning (Donahue et a Dumoulin l, 2017) #Adversarially Learned Inference (Dumoulin et al, 2017)

Evaluating GANs

- Two measures that are commonly used to evaluate GANs
  - Inception score (IS): Evaluates the distribution of generated data
  - Frechet inception distance (FID): Compared the distribution of real data and generated data

Both IS and FID measure how realistic the generated data is

- Inception Score will be high (higher is better) if
  - Very few high-probability classes in each sample: Low entropy for its conditional $p(y|x)$
  - We have diverse classes across samples: Marginal $p(y)$ is close to uniform (high entropy)

IS is defined as $\exp(\mathbb{E}_{x\sim p_g}[\mathrm{KL}(p(y|x)||p(y))])$

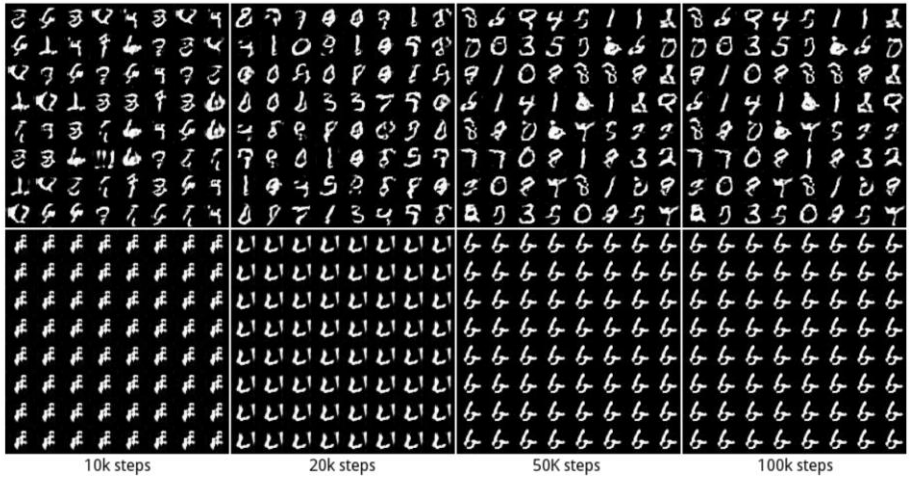
- FID uses extracted features (using a deep neural net) of real and generated data
  - Usually from the layers closer to the output layer
- These features are used to estimate two Gaussian distributions

Using real data $\mathcal{N}(\mu_R, \Sigma_\mathrm{R})$ $\mathcal{N}(\mu_G, \Sigma_\mathrm{G})$ Using generated data

- FID is then defined as $\mathbf{FID} = |\mu_G - \mu_R|^2 + \ \mathbf{trace}(\Sigma_G + \Sigma_R - (\Sigma_G \Sigma_R)^{1/2})$
  - Lower FID is better

GAN: Some Issues/Comments

- GAN training can be hard and the basic GAN suffers from several issues
- Instability of training procedure
- Mode Collapse problem: Lack of diversity in generated samples
  - Generator may find some data that can easily fool the discriminator
  - It will stuck at that mode of the data distribution and keep generating data like that



GAN 1: No mode collapse (all 10 modes captured in generation)

GAN 2: Mode collapse (stuck on one of the modes)

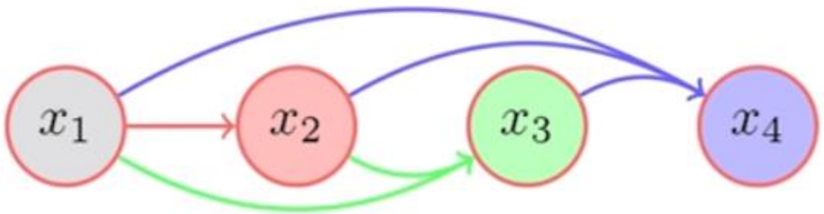
- Some work on addressing these issues (e.g., Wasserstein GAN, Least Squares GAN, etc)
- Theoretical properties of GANs yet not fully well-understood (active area of research)

Summary

- Looked at various methods for generative modeling for unsupervised learning
  - Classical methods (FA, PPCA, other latent factor models, topic models, etc)
  - Deep generative models (VAE, GAN)
- Many of these methods can also be extended to model sequential data
- There are also generative models that do not use latent variables
  - Can still be used to generate data and learn the underlying data distribution

Assuming each observation is n-dimensional

$$\prod_{i=1}^{n} p(x_i|\mathbf{x}_{<k})$$



An auto-regressive model

Can use a neural network to learn (parameters of) each of these distributions

An example: Neural Autoregressive Density Estimator (NADE)