Generative Adversarial Networks (GANs)

Idea: when it is difficult to design a good loss for reconstruction quality [recall problems with squared loss] ⇒ try to learn the loss

⇒ adversarial game with 2 players (= 2 networks)

- generator ($\stackrel{?}{=}$ decoder in AE): tries to generate images that look as realistic as possible $x \sim p_G(x)$
- discriminator (adversarial): tries to classify images into (TS) real vs. fake (generated)

  ⇒ classifier $D(x) = p(Y = \text{real} \mid X) \Leftrightarrow 1 - D(x) = p(Y = \text{fake} \mid X)$

  loss: log likelihood / cross entropy of the two classes

  $$\hat{D} = \arg\max_D \mathbb{E}_{x \sim p^*(x)} \left[\log D(x) \right] + \underbrace{\mathbb{E}_{x \sim p_G(x)} \left[\log (1 - D(x)) \right]}$$

- generator must learn to fool the discriminator = minimize

⇒ GAN loss

$$\hat{D}, \hat{G} = \arg\min_G \arg\max_D \mathbb{E}_{x \sim p^*(x)} \left[\log D(x) \right] + \mathbb{E}_{x \sim p_G(x)} \left[\log (1 - D(x)) \right]$$

generator is reparameterized by latent random numbers $z \sim p(z) = \mathcal{N}(0, \mathbb{I})$ and a deterministic network $\hat{x} = G(z)$

$$\boxed{\hat{D}, \hat{G} = \arg\min_G \arg\max_D \mathbb{E}_{x \sim p^*(x)} \left[\log D(x) \right] + \mathbb{E}_{z \sim p(z)} \left[\log (1 - D(G(z))) \right]}$$

where is the optimum of the loss?

the function ~~$a \log b +$~~ $a \log y + b \log(1-y)$ achieves its maximum at $y = \frac{a}{a+b}$ ($a \to$ output of ideal discriminator D^*)

inserting into the loss gives $\int \left[p^*(x) \log D^*(x) + p_G(x) \log(1-D^*(x)) \right] dx$

$$\text{loss}(G) = \mathbb{E}_{x \sim p^*(x)} \left[\log D^*(x) \right] + \underbrace{\mathbb{E}_{z \sim p(z)}}_{x \sim p_G(x)} \underbrace{\left[\log(1 - D^*(G(z))) \right]}_{\log(1 - D^*(x))}$$

$$= \mathbb{E}_{x \sim p^*(x)} \left[\log \frac{p^*(x)}{p^*(x) + p_G(x)} \right] + \mathbb{E}_{x \sim p_G(x)} \left[\log \frac{p_G(x)}{p^*(x) + p_G(x)} \right]$$

$$= \underbrace{KL\left(p^*(x) \,\Big\|\, \frac{p^*(x)+p_G(x)}{2}\right) + KL\left(p_G(x) \,\Big\|\, \frac{p^*(x)+p_G(x)}{2}\right)}_{\geq 0} - \log 4$$

$$\geq - \log 4$$

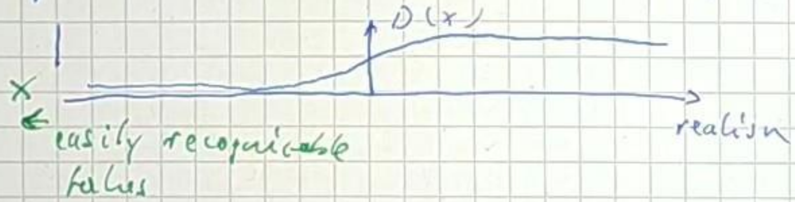
The minimum $-\log 4$ is achieved if and only if $p^*(x) = p_G(x)$ (then the two KLs are 0)

$\Rightarrow$ in the optimum, the generator learns the true data distribution $p_G(x) = p^*(x)$ and discriminator cannot distinguish fakes from reals

$\Rightarrow$ currently the state-of-the-art in image generation

· How to train GANs?

- training GANs is harder than classification networks ~ diverge easily when architecture or hyperparameters are not properly chosen
- if the discriminator is too good (relative to generator), $D(x \in \text{fake}) \approx 0$



$\Rightarrow \nabla D(x \in \text{fakes}) \approx 0$ (flat part of sigmoid)

$\Rightarrow$ don't get useful training signal for G

It is unclear in which direction we should move the parameters of G to improve.

tricks to solve:

- train D and G jointly, so that they are always about equally competent,

  alternating optimization: initialize D and G randomly

  for each minibatch: - apply k iterations to improve D $(k = 1, \dots, 4)$

  - apply 1 iteration to improve G

- use non-saturating loss:

$$\hat{D} = \arg\max_{D} \mathbb{E}_{p^*(x)} \left[\log D(x) \right] + \mathbb{E}_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right]$$

$$\hat{G} = \arg\min_{G} \mathbb{E}_{z \sim p(z)} \left[\underbrace{- \log D(G(z))}_{\text{replaces } \log(1 - D(G(z)))} \right]$$

does not suffer as much from the saturating effect (very ~~wide~~ much on the left of sigmoid)

- conditional variant (cGAN): specify attributes y that the generated image should have (e.g. faces: gender, hair color, age, ...):
  add y to the input of G: $G(z, y)$
  add a second discriminator: checks that the attributes are fulfilled
- important variant: <u>Wasserstein GAN</u> (WGAN)
  - hope: training simpler, more stable
  - idea: standard GAN discriminator learns $D(x) = p(Y=\text{real} \mid x) \in [0, 1]$
    WGAN: $\tilde{D}(x) = \log \dfrac{p(Y=\text{real} \mid x)}{p(Y=\text{fake} \mid x)} \in (-\infty, \infty)$
  - WGAN loss (naive version)
    $$\hat{\tilde{D}}, \hat{G} = \underset{G}{\arg\min}\ \underset{\tilde{D}}{\arg\max}\ \mathbb{E}_{x \sim p^*(x)}\left[\tilde{D}(x)\right] - \mathbb{E}_{z \sim p(z)}\left[\tilde{D}(G(z))\right]$$
    $+ \text{regularize}(\tilde{D})$

    does not yet work, because training can cheat: if $\mathbb{E}_{\text{reals}}[\tilde{D}] > \mathbb{E}_{\text{fakes}}[\tilde{D}]$
    $\Rightarrow$ $\tilde{D}$ could just scale the parameters of the final layer to make the difference arbitrary big

    standard solution: restrict the gradient norm of $\tilde{D}$: $\|\nabla_x \tilde{D}(x)\|_2 \leq 1$
    (name "Wasserstein" comes from relation of this constraint with Wasserstein distance, but I do not believe that this connection explains WGAN behavior)
    alternative regularization: ~~$Var(D$~~ $Var_{p^*(x)}(D(x))$ and $Var_{p_G(x)}(D(x)) \leq 1$
    hard to optimize: additional minibatch of (real, fake)-pairs, create a random point on the connection line between each pair, gradient descent of $(1 - \|\nabla_x D(x)\|_2)^2$ at these points