Lecture 23: Generative Adversarial Networks
Reference: Goodfellow et al. (2014)

Mark Hasegawa-Johnson

University of Illinois

ECE 417: Multimedia Signal Processing, Fall 2020

Outline

VAE vs. GAN

- A **VAE**
  - scores $q(z|x)$ w.r.t. predefined prior $p(z)$,
  - generates latent variables from $q(z|x)$,
  - scores data using learned generator $p(x|z)$.
- A **GAN**
  - generates latent variables from predefined prior, $p(z)$,
  - generates data using learned generator $x = G(z)$,
  - scores data using learned discriminator $D(x)$.

VAE vs. GAN

- **Prior:** Same. Both VAE and GAN assume a unit-normal Gaussian or uniform prior for z.
- **Generator:** Similar. GAN generates x from z using $x = G(z)$, therefore x must be continuous. VAE computes $p(x|z)$, so x could be either discrete or continuous.
- **Scoring:** Very different. VAE trains $q(z|x)$ to minimize $D_{KL}(p(z)\|q(z|x))$. GAN trains $D(x)$ for no purpose other than scoring x.

What is the discriminator?

- The main innovation in GAN is the discriminator, $D(x)$.
- It outputs one number, $D(x) \in (0, 1)$.
- If x is good, $D(x) \to 1$
- If x is bad, $D(x) \to 0$

How can you train the discriminator?

The discriminator is trained by giving it 50% real data, and 50% data generated synthetically by $G(x)$. Its training objective is:

- If x is real data, the discriminator wants to output $D(x) \rightarrow 1$
- If x is synthetic data generated by $G(z)$, the discriminator wants to output $D(x) \rightarrow 0$

Outline

Probabilistic interpretation of the discriminator

Let's say $y = 1$ if a token is real data, $y = 0$ if a token is fake data. The discriminator computes

$$D(x) = \Pr\{y = 1|x\}$$

Its goal is to maximize

$$\begin{aligned} V(D,G) &= \mathbb{E}_{x\in\text{data}}\left[\ln \Pr\{y=1|x\}\right] + \mathbb{E}_{x\in\text{fake}}\left[\ln \Pr\{y=0|x\}\right] \\ &= \mathbb{E}_{x\sim p_{data}}\left[\ln D(x)\right] + \mathbb{E}_{z\sim p(z)}\left[\ln\left(1 - D(G(z))\right)\right] \end{aligned}$$

Two-player minimax game

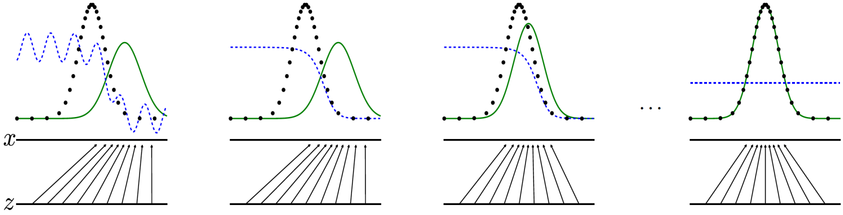
- The discriminator wants to discriminate real vs. fake data.
- The generator wants to make fake data that is as realistic as possible. So its goal is to generate data, $x = G(z)$, in order to maximize $D(G(z))$.
- D wants to maximize, and G to minimize,

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

The generator computes an implicit pdf, $p_g(x)$

- The VAE explicitly computes $p(x|z)$.
- The GAN generates z from $p(z)$, then generates $x = G(z)$. The resulting x has some pdf, that you should be able to compute using ECE 313 methods if you wanted to. Let's call this pdf $p_g(x)$.
- The goal of the generator might be phrased as follows: learn $G(x)$ so that $p_g(x)$ matches the true data distribution, $p_{data}(x)$, as well as possible.

The training process for a GAN: $D(x)$, $p_{data}(x)$, and $p_g(x)$ (c) Goodfellow et al., (2013), Figure 1



(a) (b) (c) (d)

- Blue small dots: $D(x)$
- Black large dots: $p_{data}(x)$
- Green solid: $p_g(x)$

Outline

Nash Equilibrium

- Suppose two players, D and G, are playing a game.
- Depending on their actions, they receive rewards $V_D(D, G)$ and $V_G(D, G)$, respectively (in our case, $V_G = -V_D$, but that need not be true in general).
- Each of them has perfect knowledge about the other's actions: each knows, in advance, what the other will do.

Nash Equilibrium

- Player D is called "rational" if, given knowledge of player G's action, their action is $D = \arg\max V_D(D, G)$.
- Player G is called "rational" if, given knowledge of player D's action, their action is $G = \arg\max V_G(D, G)$.
- A **Nash equilibrium** is a set of actions (D, G) such that, each player knowing in advance the other player's action, neither player has any rational incentive to change.

Nash Equilibrium for the GAN: Player D

First, suppose G is known, therefore $p_G(x)$ is known. Now D wants to maximize:

$$\begin{aligned} V(D,G) &= \mathbb{E}_{x\sim p_{data}}\left[\ln D(x)\right] + \mathbb{E}_{x\sim p_g}\left[\ln\left(1-D(x)\right)\right] \\ &= \int \left(p_{data}(x)\ln D(x) + p_g(x)\ln\left(1-D(x)\right)\right)dx \\ &= \int f(x)dx \end{aligned}$$

So for any particular x, the discriminator wants to maximize:

$$f(x) = p_{data}(x)\ln D(x) + p_g(x)\ln\left(1-D(x)\right)$$

Nash Equilibrium for the GAN: Player D

$$f(x) = p_{data} \ln D + p_g \ln (1 - D)$$

$$\frac{df}{dD} = \frac{p_{data}}{D} - \frac{p_g}{1 - D}$$

$$\frac{d^2 f}{dD^2} = -\frac{p_{data}}{D^2} - \frac{p_g}{(1 - D)^2}$$

We find the maximizer by setting $df/dD = 0$, which gives us

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

Furthermore, we see that $f(x)$ is a convex function of D, and therefore $D_G^*(x)$ is the unique global maximizer, because

$$\frac{d^2 f}{dD^2} < 0 \ \ \forall D \in [0, 1], \ p_{data} > 0, \ \ p_g > 0$$

Nash Equilibrium for the GAN: Player G

Now, let G try to win. First, let's suppose that $D(x)$ is fixed. In that case, what is the optimum strategy for G?

$$\begin{aligned} G_D^*(z) &= \arg\min \mathbb{E}_{z\sim p(z)}\left[\ln\left(1 - D(G(z))\right)\right] \\ &= \arg\max D(G(z)) \end{aligned}$$

In other words, $G(z)$ should always output the same x (the one that maximizes $D(x)$), regardless of what z is! Though that's a good strategy for player G, it's not a very good machine learning result.

Avoiding the trivial solution

- How can we avoid the trivial solution, where $G(z)$ always outputs the same x?
- Answer: we have to re-train $D(x)$. If $G(z)$ always outputs the same x, then the probability density goes to infinity ($p_g(x) \rightarrow \infty$) for that token. If $D(x)$ is allowed to respond rationally, then it will penalize that over-sampled token:

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \rightarrow_{p_g(x)\rightarrow\infty} 0$$

Nash Equilibrium for the GAN: Player G

In order to get a better machine learning result, let's assume that whatever strategy G uses, D will choose the optimal counter-strategy $D_G^*(x)$. Therefore, G wants to choose p_G in order to minimize

$$V(D_G^*, G) = \mathbb{E}_{x\sim p_{data}}\left[\ln D_G^*(x)\right] + \mathbb{E}_{x\sim p_g}\left[\ln\left(1 - D_G^*(x)\right)\right]$$

$$= \mathbb{E}_{x\sim p_{data}}\left[\ln \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right] + \mathbb{E}_{x\sim p_g}\left[\ln \frac{p_g(x)}{p_{data}(x)+p_g(x)}\right]$$

$$= -\ln(4) + D_{KL}\left(p_{data}\middle\|\frac{p_{data}+p_g}{2}\right) + D_{KL}\left(p_g\middle\|\frac{p_{data}+p_g}{2}\right)$$

The KL divergence $D_{KL}(p\|q)$ is a concave function of both p and q. Among p and q that are pdfs, it has a unique global minimizer at

$$p_g^*(x) = p_{data}(x)$$

What we've proved

- **Proven:** For *any* generator G, the value function $V(D, G)$ is a convex function of D, with a unique global maximizer

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

- **Proven:** If G is updated in a series of gradient steps, and if D has time to converge to D_G^* in between each pair of G-steps, then G will converge to the unique global minimizer of $V(D_G^*, G)$:

$$p_g^*(x) = p_{data}(x)$$

Mode collapse

- **Something not true:** it is not true that, for a fixed $D(x)$, we can perform multiple gradient steps on $G(x)$. If $D(x)$ can be easily fooled, then $G(z)$ will converge to an incorrect pdf that fools it.
- **Mode collapse:** Often, if $D(x)$ doesn't know the data well, there's a particular x^* that always fools it. $G(z)$ can "win" by always producing the same output:

$$G(z) \rightarrow x^* \quad \text{if} \quad D(x^*) = 1$$

- Mode collapse can be avoided by training $D(x)$ to convergence between each pair of G-steps, so that misguided G-updates are corrected before they get too bad. If mode collapse happens, though, it may be hard to recover.

Outline

Summary

- A GAN is a pair of networks $G(z)$ and $D(x)$ s.t.

$$\min_G \max_D \mathbb{E}_{x\sim p_{data}}\left[\ln D(x)\right] + \mathbb{E}_{z\sim p_z}\left[\ln\left(1 - D(G(z))\right)\right]$$

- If $D(x)$ is trained to convergence between each pair of G-steps, the GAN will reach the global Nash equilibrium

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

- If $G(z)$ is allowed to converge while $D(x)$ is incorrect, it will lead to mode collapse. In order to avoid mode collapse, D needs to converge enough, between G-steps, so that it reverses the gradient near the bad mode, pushing G away from x^*.