Generative Adversarial Nets

Ruyue Han

August 25, 2018

1. Adversarial nets

The adersarial nets include two parts: generator G and discriminator D,they are all neural networks and can be changed following your mind.The purpose of G is to create images ,the D is to identify the source of images.In other words,D and G play the folloing two-player minimax game with value function V(G,D).

$$\begin{aligned}\min_G \max_D V(G,D) &= \mathbb{E}_{x\backsim p_{data}(x)}\left[\log D(x)\right] + \mathbb{E}_{z\backsim p_z(z)}\left[\log 1 - D(G(z))\right] \\ &= \mathbb{E}_{x\backsim p_{data}(x)}\left[\log D(x)\right] + \mathbb{E}_{x\backsim p_g(x)}\left[\log 1 - D(x)\right]\end{aligned} \tag{1}$$

where $p_{data}(x)$ is the real data's distribution, and $p_z(z)$ is the noise's distribution which can be changed, $p_g(x)$ is generators distribution.

The final purpose is that G can create realistic image and D can't identify where the image is coming from,which means the generator distribution p_g is closed to real data distribution P_{data}.The training steps can be see as Fig.1

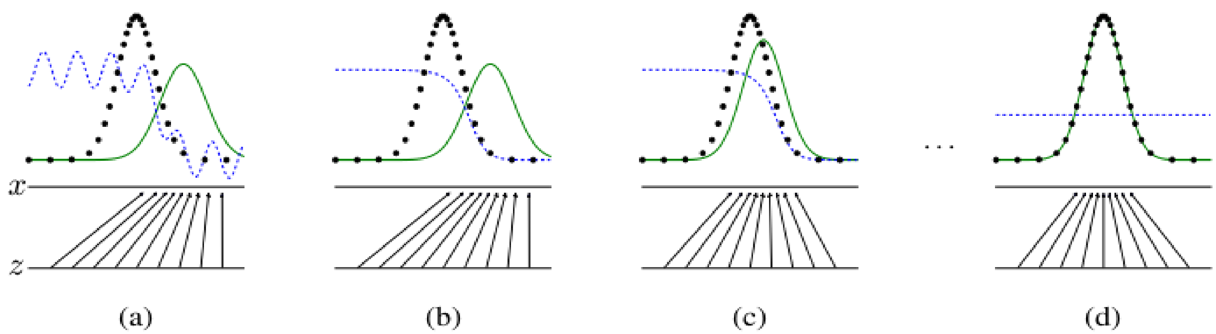

Figure 1. Generative adversarial nets are trained by simultaneously updating the discriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) p_{data} from those of the generative distribution p_g (G) (green, solid line).

2. Theoretical Results

2.1. Global Optimality of $p_g = p_{data}$

Author first consider the optimal discriminator D for any given generator G.

$$\begin{aligned}V(G,D) &= \mathbb{E}_{x\backsim p_{data}(x)}\left[\log D(x)\right] + \mathbb{E}_{x\backsim p_g(x)}\left[\log 1 - D(x)\right] \\ &= \int_x p_{data}(x)\log\left(D(x)\right)dx + \int_x p_g(x)log(1-D(x))dx \\ &= \int_x p_{data}(x)\log\left(D(x)\right) + p_g(x)log(1-D(x))dx\end{aligned} \tag{2}$$

For any $(a,b) \in \mathbb{R}^2$,the function $f(x) = alog(x) + blog(1-x), 0 < x < 1$,it's maximum is $\frac{a}{a+b}$.

Proof:the f(x) is differentiable function,when $0 < x < 1$,and we know when the derivative function $f^{'}(x) = 0$,the value of f(x) is maxmun.

$$f^{'}(x) = \frac{a}{x} + \frac{-b}{1-x} = 0 \tag{3}$$

so the answer is $\frac{a}{a+b}$. Now we can get the optimal discriminator D is

$$D^{*}(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{4}$$

and at this time

$$\begin{aligned}
\max_D V(G,D) &= V(G,D^*) = \mathbb{E}_{x \backsim p_{data}(x)}\left[\log D^*(x)\right] + \mathbb{E}_{x \backsim p_g(x)}\left[\log\left(1 - D^*(x)\right)\right] \\
&= \mathbb{E}_{x \backsim p_{data}(x)}\left[\log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}\right] + \mathbb{E}_{x \backsim p_g(x)}\left[\log \frac{p_g(x)}{p_{data}(x) + p_g(x)}\right] \\
&= \ldots \\
&= -\log 4 + KL(p_{data}(x)||\frac{p_{data}(x) + p_g(x)}{2}) + KL(p_g(x)||\frac{p_{data}(x) + p_g(x)}{2}) \\
&= -\log 4 + 2JS(p_{data}(x)||p_g(x))
\end{aligned} \tag{5}$$

Given different G we got different JS distance,and Our purpose is to get the shortest JS distence.So only and only $p_{data} == p_g$,the generative model perfectly replicate the data generating process.

In the practice, we update the discriminator by ascending its stochastic gradien and update the generator by descending its stochastic gradient.

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

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

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Figure 2. The algorithm of minmax game.

3. Theoretical Results