# GAN collections

Zhi Li

May 28, 2019

# 1 GAN

## 1.1 Applications

GAN is kind-of unsupervised learning for generating sequences, graphs, videos etc. Check out all the applications here. collections of awesome GAN applications. Gernerally, input some random noise, the model will be trying to output the data satisfying to your domain. Some interesting examples like horse to zebra, shoes transformation, and face aging animation.

Problems lies in the aera that how to transform the random input towards something meaningful. This corresponds to the data distribution transformation, which will later display in the loss function.

The philosophy behind GAN is that, we train two frameworks, one for generator which input random noise and output useful results, and the other is discriminator which will drive generator to produce good results. This framework corresponds to a minimax two-player game. Generator will be forcing to generate something to fool discriminator while discriminator will classify the fake data.

Some related work to do so including deep-Boltzmann machine, Deep Belief Networks, Variational AutoEncoder (VAE)

## 1.2 Adversarial Nets

In his paper, he suggests that optimal Discriminator $D$ is achieved when $G$ is fixed.

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{1}$$

**Theorem**. If $p_g = p_{data}$ which means the global minimum. C(G) will achieve the value of $-log4$.

# 2 Various GAN

## 2.1 MMGAN

The one proposed by Goodfellow, the original loss function used for discriminator is $-log(1-D(x))$ (cross-entropy)

**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 1: Algorithm description

## 2.2 NSGAN

In NSGAN, the difference between it with MMGAN is that it starts to use $-log(D(x))$ because it proves that, at the begining, the gradient for $-log(1 - D(x))$ is small which may not reasonable as we need the gradient to be large to converge. So with $log(D(x))$, it initialized a higher gradient at first and decreasing progressively.

| GANS | Archive | Loss Function | |
|---|---|---|---|
| GAN | Arxiv | $L_D^{GAN} = E[log(D(x))] + E[log(1 - D(G(z))] L_G^{GAN} = E[log(D(G(z)]$ | |