**SUMMARY** **Generative Adversarial Networks(2014)**

Adversary = opponent, enemy

GAN are deep neural net architectures comprising of 2 neural networks competing against each other. They are CNN or simple Vanilla networks.

GAN are neural networks that are trained in an adversarial (competitive) manner **to generate data mimicking some distribution.**

GAN consists of :
**Discriminative Model -** Classfication problem
**Generative Model –** Uses Unsupervised learning approach, new examples are similar and indistinguished from input data.

So basically Generator network: try to fool the discriminator by generating real-looking images.

Discriminator network: try to distinguish between real and fake images.

In GAN, both models are trained using highly successful backpropagation and dropout algorithms and sample from the generative model using only forward propagation. No approximate inference or Markov chains are necessary.

The adversarial modeling framework is most straightforward to apply when the models are both
multilayer perceptrons.

$P_g$ = Generator's distribution
$P_z(z)$= Noise Varible
G(z;theta$_g$) = Mapping to data space
G is a differentiable function represented by multilayer perceptron
D(x;theta$_d$) = 2nd Multilayer Perceptron

D assigns correct label to training examples as well as samples from G

Objective Function alternates between:

1. Gradient ascent on discriminator

$$\max_D V(D) = \underbrace{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})]}_{\text{recognize real images better}} + \underbrace{\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]}_{\text{recognize generated images better}}$$

Discriminator wants to maximize objective such that D(x) is close to 1 (real) and D(G(z)) is close to 0 (fake)

2. Gradient descent on generator

$$\min_G V(G) = \underbrace{\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]}_{}$$

Optimize G that can fool the discriminator the most.

Generator wants to minimize objective such that D(G(z)) is close to 1(discriminator is fooled into thinking generated G(z) is real)

The discriminator usually wins early against the generator. It is always easier to distinguish the generated images from real images in early training

Training a GAN

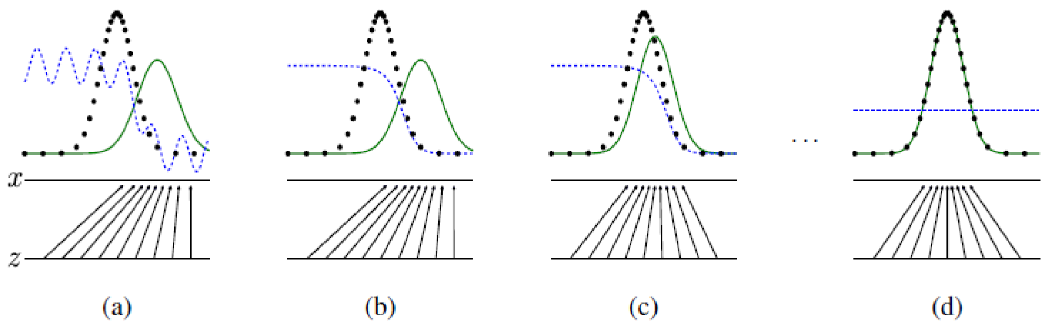
1. Train the discriminator then train the Generator.

1. Define the problem
2. Choose architecture of GAN
3. Train Discriminator on real data/ **Epoch**
4. Generate Fake inputs for Generator
5. Train Discriminator on Fake Data
6. Train Generator with the output of Discriminator

Here is the pseudo code that puts all things together and shows how GAN is trained.

**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 $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
        • Sample minibatch of $m$ examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
        • Update the discriminator by ascending its stochastic gradient:
              $$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right].$$
    end for
    • Sample minibatch of $m$ noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
    • Update the generator by descending its stochastic gradient:
          $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right).$$
end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.
```



Black dots: Generating distribution, $P_x$

Green solid: Generative distribution, $P_g(G)$

Blue dashed: Discriminative distribution, D

(a) Poorly model fit
(b) After updating D
(c) After updating G
(d) Mixed strategy equilibrium

Proofs:

Proposition 1. For G fixed, the optimal discriminator D is

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

Proof. The training criterion for the discriminator D, given any generator G, is to maximize the
quantity V (G;D)

$$\begin{aligned} V(G, D) &= \int_x p_{\text{data}}(x) \log(D(x))dx + \int_z p_z(z) \log(1 - D(g(z)))dz \\ &= \int_x p_{\text{data}}(x) \log(D(x)) + p_g(x) \log(1 - D(x))dx \end{aligned}$$

For any (a; b) 2 R2 n f0; 0g, the function y ! a log(y) + b log(1 □ y) achieves its maximum in
[0; 1] at a/(a+b).
Modify the loss function

$$\begin{aligned} C(G) =& \max_D V(G, D) \\ =& \mathbb{E}_{x \sim p_{\text{data}}}[\log D_G^*(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D_G^*(G(z)))] \\ =& \mathbb{E}_{x \sim p_{\text{data}}}[\log D_G^*(x)] + \mathbb{E}_{x \sim p_g}[\log(1 - D_G^*(x))] \\ =& \mathbb{E}_{x \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(x)}{P_{\text{data}}(x) + p_g(x)}\right] + \mathbb{E}_{x \sim p_g}\left[\log \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)}\right] \end{aligned}$$

Find the optimal value for $\boldsymbol{V}$:

$$\begin{aligned}\min_G V(D^*, G) &= \int_x \Big(p_r(x)\log D^*(x) + p_g(x)\log(1 - D^*(x))\Big)dx \\ &= \int_x \Big(p_r(x)\log\frac{p_r(x)}{p_r(x)+p_g(x)} + p_g(x)\log\frac{p_g(x)}{p_r(x)+p_g(x)}\Big)dx\end{aligned}$$

$$\begin{aligned}D_{JS}(p_r\|p_g) &= \frac{1}{2}D_{KL}(p_r\|\frac{p_r+p_g}{2}) + \frac{1}{2}D_{KL}(p_g\|\frac{p_r+p_g}{2}) \\ &= \frac{1}{2}\Big(\int_x p_r(x)\log\frac{2p_r(x)}{p_r(x)+p_g(x)}dx\Big) + \frac{1}{2}\Big(\int_x p_g(x)\log\frac{2p_g(x)}{p_r(x)+p_g(x)}dx\Big) \\ &= \frac{1}{2}\Big(\log 2 + \int_x p_r(x)\log\frac{p_r(x)}{p_r(x)+p_g(x)}dx\Big) + \\ &\quad \frac{1}{2}\Big(\log 2 + \int_x p_g(x)\log\frac{p_g(x)}{p_r(x)+p_g(x)}dx\Big) \\ &= \frac{1}{2}\Big(\log 4 + \min_G V(D^*, G)\Big)\end{aligned}$$

i.e.

$$\min_G V(D^*, G) = 2D_{JS}(p_r\|p_g) - 2\log 2$$

Since the Jensen–Shannon divergence between two distributions is always non-negative and zero only when they are equal,
Hence the only solution is pg = pdata(i.e. pr).......

***With $p = q$, the optimal value for D and V is***

$$D^*(x) = \frac{p}{p+q} = \frac{1}{2}$$

$$\begin{aligned}\min_G \max_D V(D, G) &= \mathbb{E}_{x\sim p_r(x)}[\log\frac{1}{2}] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - \frac{1}{2})] \\ &= -2\log 2\end{aligned}$$

Disadvantages of GAN

1. There is no explicit representation of pg(x)
2. D must be synchronized well with G during training (in particular, G must not be trained too much without updatingD, in order to avoid “the

Helvetica scenario” in which G collapses too many values of z to the same value of x to have enough diversity to model pdata),

Advantages of GAN

1. Markovchains are never needed, only backprop is used to obtain gradients, no inference is needed during learning, and a wide variety of functions can be incorporated into the model.
2. Adversarial models also gain statistical advantage from the generator network not being updated directly with data examples, but only with gradients flowing through the discriminator. This means that components of the input are not copied directly into the generator’s parameters.
3. Another advantage of adversarial networks is that they can represent very sharp, even degenerate distributions, while methods based on Markov chains require that the distribution be somewhat blurry in order for the chains to be able to mix between modes.