Generative Adversarial Nets

→ G, D

G → Generator → pass random noise through a multilayer perceptron

D → Discriminator

→ Doesn't require Markov chains for sampling.

→ Train G to maximise $\log(1 - D(G(z)))$

→ Objective :- min for G
max for D

$V(D, G)$ → value function
→ evaluates performance of GAN

$E_{x \sim p_{data}(x)}$ → expectation over real data samples x drawn from true data distribution $p_{data}(x)$.
→ Avg. over real data samples

$\log D(x)$ → Log of D's output when given a real data sample x.

$E_{z \sim p_z(z)}$ → expectation over random noise z from prior distribution $p_z(z)$.
→ Avg. over random noise samples.

$\log(1 - D(G(z)))$ → Log of D's output when given a generated sample $G(z)$, z being a random

Date
Page

→ Optimising D ~~over~~ to completion in inner loop is computationally prohibitive, and on finite datasets - overfitting occurs.

→ $D(x) = \dfrac{P_{data}(x)}{P_{data}(x) + P_g(x)}$

At the end, $D(x) = \frac{1}{2}$ as $P_{data}(x) = P_g(x)$

(1) → $$\boxed{\min_G \max_D V(D,G) = E_{x \sim P_{data}(x)}[\log(D(x))] + E_{z \sim P_z(z)}[\log(1 - D(G(z)))]}$$

→ Global optimum for $P_g = P_{data}$

→ Algorithm-1

for no. of training iterations {
    for k steps {
        → take m noise samples from $P_g(z)$
        → take m samples from P_{data} (→ real data)
        → Update discriminator by ascending its stocastic gradient :-
        (as based on a subset not entire set)

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log(D(x^i)) + \log(1 - D(G(z^i)))\right]$$

$$\left(\frac{\partial}{\partial \theta_1} + \frac{\partial}{\partial \theta_2} \cdots\right)$$

}

P.T.O.

→ take m noise samples from $p_g(z)$
→ Update G by descending stocastic gradient:–

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z^i))) :$$

}

* **Finding Optimal Discriminator D :–**

→ $V(G, D) = E_{x \sim p_{data}(x)}[\log(D(x))] +$

$$E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

→ $E[x] = \sum x_i p(x_i)$ (Σ is same as $\int dx$)

→ $V(G, D) = \int_x p_{data}(x) \log(D(x))\,dx + \int_z p_z(z) \log(1 - D(g(z)))\,dz$

→ We can bring everything to a single variable as:–

$$I = \int f(a)\,da + \int f_1(b)\,db = \int (f(a) + f_1(a))\,da.$$

→ $V(G, D) = \int_x \left(p_{data}(x) \log(D(x)) + p_z(z) \log(1 - D(G(x)))\right)dx$

→ Consider the function :–

$y \to a \log(y) + b \log(1-y)$, $(a,b) \in \mathbb{R}^2 \setminus \{0,0\}$

or $y = a \log(x) + b \log(1-x)$

$$y' = \frac{a}{x} + \frac{b}{1-x}(-1)$$

$y' = 0$ at $\frac{a}{\ } = \frac{b}{\ }$

$a - ax = bx$

$\therefore x = \frac{a}{a+b}$.

$y'' = -\frac{a}{x^2} - \frac{b}{(1-x)^2}$

as $x \in (0,1)$, y'' is always -ve.

$\therefore x = \frac{a}{a+b} \rightarrow$ maxima.

→ It can be proved that this is global maxima (end point value method).

→ Thus, optimal $D^*_G(x) = \frac{P_{data}(x)}{P_{data}(x) + P_g(x)}$

→ $C(G) = \max_D (V(G, D))$

$= E_{x \sim P_{data}} [\log(D^*_G(x))] + E_{z \sim P_z} [\log(1 - D^*_G(G(z)))]$

$= E_{x \sim P_{data}} [\log(D^*_G(x))] + E_{x \sim P_g} [\log(1 - D^*_G(x))]$

$= E_{x \sim P_{data}} \left[\log\left(\frac{P_{data}(x)}{P_{data}(x) + P_g(x)}\right)\right] +$

$E_{x \sim P_g} \left[\log\left(\frac{P_g(x)}{P_{data}(x) + P_g(x)}\right)\right]$

→ Global minimum of $C(G)$ is achieved when $P_g = P_{data}$

$C(G) = E_{x \sim P_{data}} [\log(\frac{1}{2})] + E_{x \sim P_g} [\log(\frac{1}{2})]$

$= -\log 2 - \log 2$

$= -\log 4$

$(E_x[c] = c$, $c =$ constant$)$

→ Further,

$$C(G) = -\log(4) + 2\,JSD(P_{data} \| P_g)$$

where,

$$2\,JSD = KL(P \| M) + KL(Q \| M), \quad M = \frac{P+Q}{2}$$

2 JSD → Jensen-Shanon Divergence

KL → Kullback-Leibler Divergence. → $\boxed{KL(P \| Q) = \int p(x) \log\left(\frac{P(x)}{Q(x)}\right) dx}$

JSD → non-negative.

∴ $C(G)$ → global min. when $2\,JSD(P_{data} \| P_g) = 0$

→ occurs when $P_{data} = P_g$

→ At $P_g = P_{data}$ → model generates data that perfectly replicates P_{data} (real data)

* Experiment :-

→ Please Refer the code for generating MNIST like images using GAN.

* Advantages :-

→ No Markov Chains (∵ dep. on prev. state)

→ Computational advantage as only gradients flow through D.

* Disadvantages :-

→ "Helvetica scenario" may occur, G collapses too many values of z to the same value of x. (To avoid, G must not be trained too much without updating D).