Generative Adversarial Networks (GANs)

24 May 2022 15:52

→ We will skip the introduction part as you can find it anywhere on the internet.

AIM: Given some data, say images, a trained GAN will mimick the data and produce realistic looking images.

NOTE:

① Input to a trained GAN is just a random noise vector of any dimension (which should be the same dimensions the Generator was trained with).

② The Generator never really 'sees' the actual/real data for generating images. Thus, training the Generator is often slower and/or harder than training the Discriminator.

③ The Discriminator is just a binary image classifier, which is simultaneously trained from scratch along with the Generator.

④ A useful tip for implementing GANs is to NOT to assume that the Generator model is not enough capable / complex as compared to the Discriminator if you see the Discriminator loss going towards 0 and the Generator loss sky-rocketing during the initial training epochs.

MATHEMATICS:

1. If X is a Random Variable such that $X : \Omega \to \mathbb{R}$,
(Discrete)

Expectation of X, denoted by $E(X)$ is given by

$$E(X) = \sum_{x \in R(X)} x\, P(X = x)$$

$$E(X) = \sum_{x \in R(X)} x\, f_x(x) \quad \text{(discrete)}$$

$$E(X) = \int_{-\infty}^{+\infty} x\, f'_x(x)\, dx \quad \text{(continuous)}$$

Provided that $E(X)$ is absolutely convergent, i.e.

$$\sum_{x \in R(X)} |x|\, f_x(x) < \infty$$

where $f_x(x)$ is the Probability Mass Function of X.
and $f'_x(x)$ is the Probability Density Function of X.

2. Expectation of a function of a Random Variable

If $E(X) = \sum_{x \in R(X)} x\, f_x(x)$ (given it's absolutely convergent)

then if $g(x)$ is a continuous function of X, then:

$$E(g(X)) = \sum_{x \in R(X)} g(x)\, f_x(x) \quad \text{(discrete)}$$

$$E(g(X)) = \int_{-\infty}^{+\infty} g(x)\, f'_x(x)\, dx \quad \text{(continuous)}$$

Provided that $E(X)$ is absolutely convergent, i.e.

$$\sum_{x \in R(X)} |g(x)|\, f_x(x) < \infty$$

where $f_x(x)$ is the Probability Mass Function of X
and $f'_x(x)$ is the Probability Density Function of X.

(All thanks to the Probability & Statistics course!)

3. Log loss / Binary Cross Entropy loss:

$$\text{Loss} = \left[\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right]$$

$$Loss = \left[\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]$$

(I have dropped the averaging term $1/N$ and -ve sign)

This is exactly the same loss function that we use for Logistic Regression.

If we carefuly see the behavior of the BCE loss,

When $y_i = 0$: (A fake training example)

$$Loss_0 = \log(1 - \hat{y}_i)$$

Since we want to make $\hat{y}_i = 0$ as well, thus we aim at making $Loss_0 \to 0$ as our model slowly trains.

If our model makes a terrible prediction of $\hat{y}_i = 1$, then $\log(1-\hat{y}_i) \longrightarrow (\log(0) = -\infty)$
Thus, our aim is to maximize $Loss_0$.

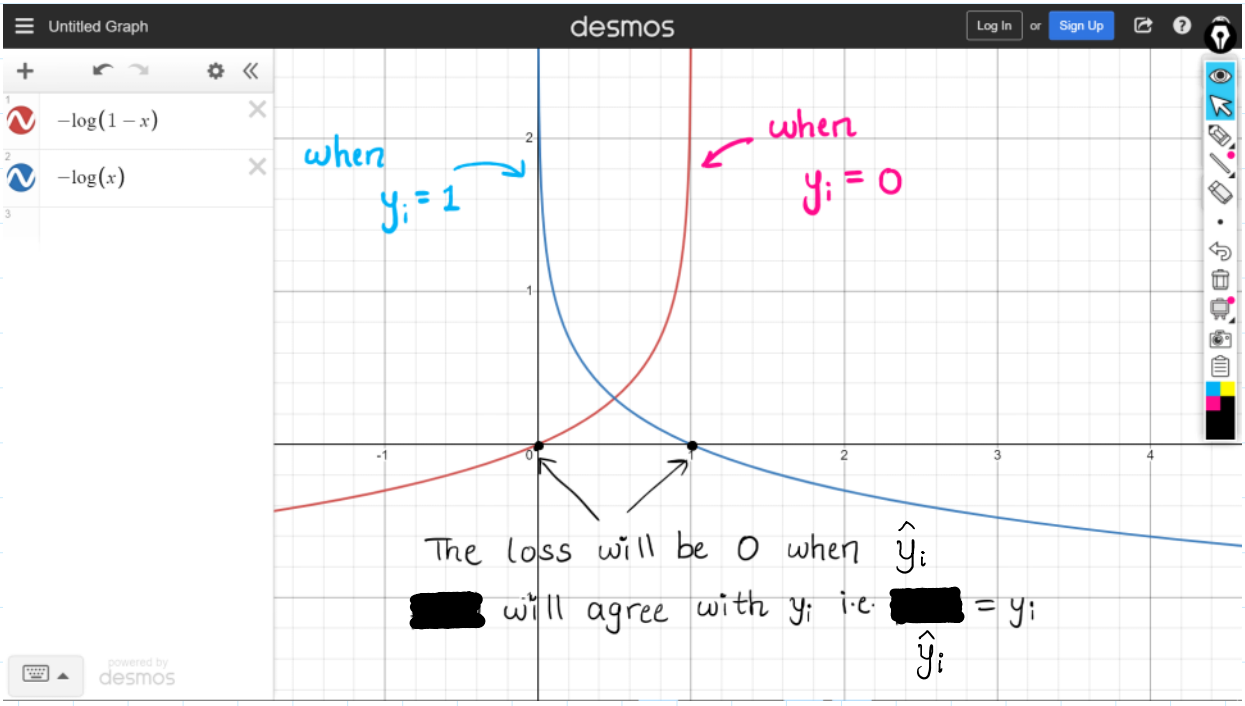
When $y_i = 1$: (A true training example)

$$Loss_1 = \log(\hat{y}_i)$$

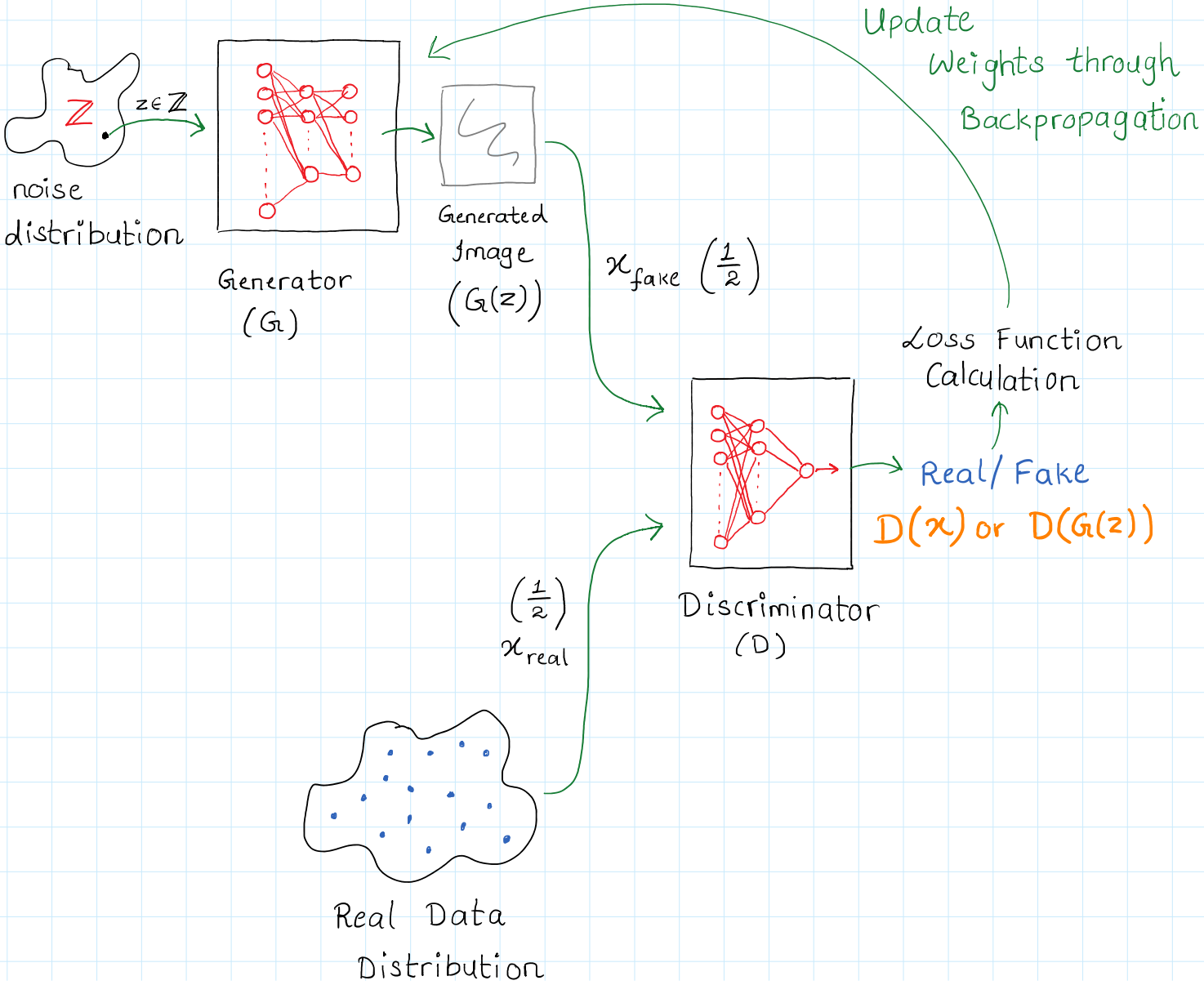
Since we want to make $\hat{y}_i = 1$ as well, thus we aim at making $Loss_1 \to 0$ as our model slowly trains.

If our model makes a terrible prediction of $\hat{y}_i = 0$, then $\log(\hat{y}_i) \longrightarrow (\log(0) = -\infty)$
Thus, our aim is to maximize $Loss_1$.

4. Getting a taste of the log loss / BCE loss.



Workflow of GAN Training :



The Original min-max Loss Function Derivation :

Given the BCE loss, removing the negative sign gives us :

$$\mathcal{L}(y_i, \hat{y}_i) = \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right]$$

we can replace $\hat{y}_i$ with $D(x)$ (The discriminator's prediction of whether x is real or fake).

When we feed a real image, $y_i = 1$

$$\mathcal{L}(y_i, \hat{y}_i) = \log(D(x_i)) \quad \text{where} \quad x_i \sim P_{real}(X)$$

$$\rightarrow \mathcal{L}(1, D(x_i)) = \log(D(x_i)) = \mathcal{L}_1 \quad - (1)$$

When we feed a fake image, $y_i = 0$

$$\mathcal{L}(y_i, \hat{y}_i) = \log(1 - D(G(z_i))) \quad \text{where} \quad z_i \sim \mathcal{Z}$$

$$\rightarrow \mathcal{L}(0, D(G(z_i))) = \log(1 - D(G(z_i))) = \mathcal{L}_0 \quad - (2)$$

The Discriminator's Story :

It wants to correctly classify the real and the fake images, thus :

$$D(x_i) = 1$$
$$D(G(z_i)) = 0 \qquad \forall \; x_i \sim P_{real}(X), \; z_i \sim \mathcal{Z}$$

This means it's trying to make $\mathcal{L}_0$ and $\mathcal{L}_1$ converge to 0 (from ① & ②).

Initially, when the discriminator performs poorly :

$$D(G(z_i)) = 1$$

or

for some $z_i \sim \mathcal{Z}$ and/or some $x_i \sim P_{real}(X)$

$D(G(z_i)) = 1$

or

$D(x_i) = 0$

for some $z_i \sim \mathbb{Z}$ and/or some $x_i \sim P_{real}(X)$

This will cause $\mathcal{L}_0$ and/or $\mathcal{L}_1$ to tend to $-\infty$.

Thus, in a nutshell, the Discriminator is trying to maximize both $\mathcal{L}_0$ and $\mathcal{L}_1$.

$$D: \max_D \left[\log(D(x_i)) + \log(1 - D(G(z_i))) \right] \quad - (3)$$

The Generator's Story:

It wants to fool the Discriminator, thus

$$D(G(z_i)) = 1 \quad \forall \; z_i \sim \mathbb{Z}$$

This means it wants to make $\mathcal{L}_0$ tend to $-\infty$. (From (2))

Initially, when the Generator performs poorly:

$$D(G(z_i)) = 0 \quad \text{for some } z_i \sim \mathbb{Z}$$

This will cause $\mathcal{L}_0$ to tend to 0.

Thus, in a nutshell, the Generator is trying to minimize $\mathcal{L}_0$.
This is the same as saying that the Generator is trying to minimize both $\mathcal{L}_0$ and $\mathcal{L}_1$ as there is no role of the generator in the value of $\mathcal{L}_1$.

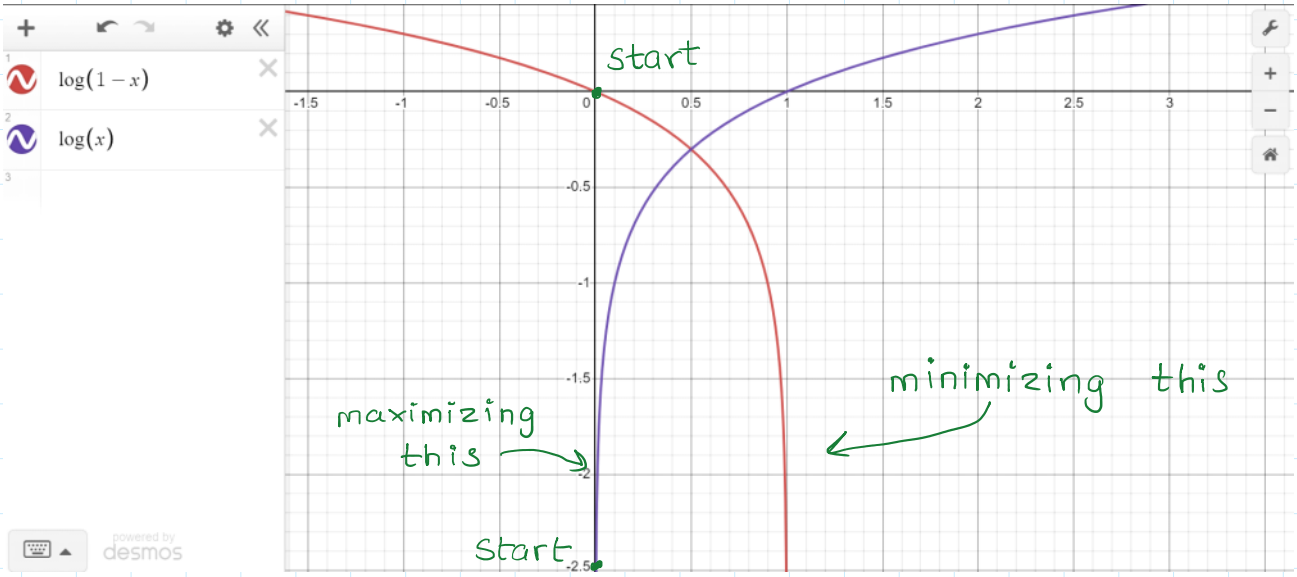
$$G: \min_G \left[\log(1 - D(G(z_i))) \right] \quad \text{(In practice)}$$

$$G: \min_G \left[\log(D(x_i)) + \log(1 - D(G(z_i))) \right] \quad - (4) \quad \text{(In theory)}$$

↑ won't make a difference as

there is no role of Generator here.

A problem with this Generator Loss is saturating gradients during the early stages of Generator training.



So instead of minimizing $\log(1 - D(G(z^{(i)}))$, we will try maximizing $\log(D(G(z^{(i)})))$.

$$G: \log(D(G(z^{(i)})))$$

Our aim : Maximize $\{G\}$

In practice, the Generator is only trained to

$$\max_G \log(D(G(z^{(i)})))$$

(Non-saturating gradients!)

From ③ and ④,

$$\min_G \max_D \left[\log(D(x_i)) + \log(1 - D(G(z_i)))\right]$$

Now this derivation was for a single input (x_i and z_i giving $D(x_i)$ and $D(G(z_i))$).

For acomodating multiple samples, we should take a weighted sum of all the samples which, converted into a probabilistic term will give us the expectation.

$$\min_G \max_D \sum_{i=1}^{N} \left[P_{data}(x) \log(D(x_i)) + P_z(z) \log(1 - D(G(z))) \right] \quad \text{(discrete)}$$

$$\min_G \max_D \int_{-\infty}^{+\infty} P_{data}(x) \log(D(x_i))\, dx + \int_{-\infty}^{+\infty} P_z(z) \log(1 - D(G(z)))\, dz \quad \text{(continuous)}$$

I can also write this probalistically as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$