Introduction to GANs

18 October 2023 06:11 PM

Generative Adversarial Networks (GAN)

→ Generative Models: These networks try to model the distribution of input.
Eg. In ML, The distribution X and y is given as input.
whereas for generative models they try to find

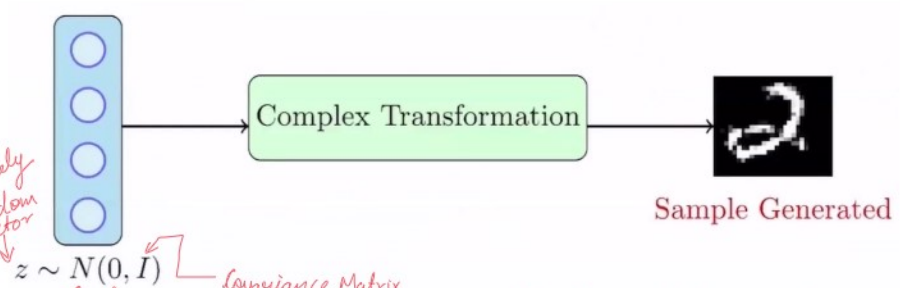
→ Discriminative Models: These networks don't exactly care about the distribution, rather they focus more on discriminating or distinguishing between the different class labels.

objective of generative models is to learn the input distribution

→ so, given some training data (say MNIST images), it comes from an underlying distribution

whereas, the only thing GANs want to do is, to draw samples from this input distribution.

In layman terms, GANs want to generate images that are similar to MNIST dataset images. It is basically trying to model the input distribution in order to generate the new images.



purely Random vector

Normal Distribution

0 mean

Covariance Matrix as we have a vector

So to generate an image:

1.) We start with some random vector.

2.) Then we try to apply some complex transformations in order to generate the image by transforming the given vector to image.

∴ Because it is difficult to sample from the input distribution, we start from a sample from normal distribution and then transform it into a sample from the input distribution

Q How to get the said "Complex Transformation"?

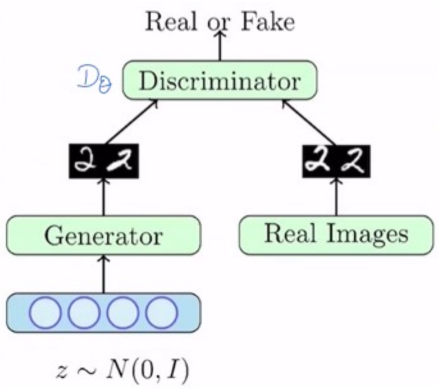
Ans By using a deep network and learn it.

Now in order to perform the above task:

We will use a two player game setup between:

1.) Generator 2.) Discriminator

{θ: parameters of the Discriminator}



{parameters of the Generator: weights and biases} ϕ

→ Job of the generator is to produce images which look so good that the discriminator is fooled.

→ Job of the discriminator is to get better at distinguishing between true images and generated images.

⇒ So it becomes a tug of war wherein the generator keeps on trying to fool the discriminator and the later tries to distinguish them.

So the generator takes $z \sim N(0, I)$ and produces $\underbrace{G_\phi(Z)}_{\text{Neural Network for the generator}} = X$ → Image

and the discriminator takes X (image) and produces a score

$D_\theta(x) \in [0, 1]$ { Similar to classification between 2 classes
→ 0 : Fake Image
→ 1 : True Image }

* In layman terms discriminator will take the generated image and the true image and based upon the score it will classify if the image is fake or real. Now initially discriminator will perform poorly i.e. since it will not be able to classify properly it will have a loss and will update its parameters to become better.

Objective Function

Generator: Given some Z, it wants to maximize

$$\max D_\theta\left[G_\phi(Z)\right]$$

Now, according to optimization theory

$$\max f(x) \approx \max g(f(x)) \quad \{\text{iff. } g() \text{ is a monotonic function}\}$$

$$\Rightarrow \max \log\left[D_\theta\left(G_\phi(Z)\right)\right]$$

Now w.k.t. $\max f(x) \equiv \min(1 - f(x))$
└ equivalent to / same as.

$$\Rightarrow \min \log\left[1 - D_\theta\left(G_\phi(Z)\right)\right]$$

Now there is a small road block in the above equation → we know $\log()$, $D_\theta()$, $G_\phi()$ but not about Z as it is a random variable, and it is not possible to maximize/minimize something that is random.
So instead we try to minimize/maximize the expected value.

$$\Rightarrow \min_\phi \underbrace{\mathbb{E}_{Z\sim N(0,I)}}_{\text{to remove the randomness of } Z} \left\{\log\left[1 - D_\theta\left(G_\phi(Z)\right)\right]\right\}$$

Discriminator: It will try do the reverse of generator.

$$\max_\theta \mathbb{E}_{Z\sim N(0,I)} \left\{\log\left[1 - D_\theta\left(G_\phi(Z)\right)\right]\right\} + \max_\theta \mathbb{E}_{X\sim True}\left\{D_\theta(X)\right\}$$

So now the overall objective function will be:

$$\boxed{\min_\phi \max_\theta \left\{\mathbb{E}_{Z\sim N(0,I)} \left\{\log\left[1 - D_\theta\left(G_\phi(Z)\right)\right]\right\} + \mathbb{E}_{X\sim True}\left[D_\theta(X)\right]\right\}}$$

for minimization : use gradient descent
for maximization : use gradient ascent } Utilized during backpropagation