Generative Adversarial Network

Atreye Ghosh(827009108)
M.S.,CEEN
Texas A&M University
aghosh32@tamu.edu

February 14, 2019

1 Maximum Mean Discrepancy

We begin by defining what an MMD does. Let us suppose we have 2 distributions P and Q. We wish to know how close or how similar these 2 distributions are. Let us say we have n samples called vector x from P and m samples called vector y from Q. The MMD states that we can have a transformation function ϕ that maps every sample into $\phi(x_i)$. Then, if we take the mean μ_P of all the n $\phi(x_i)$ in the sample set x and the same for y corresponding to μ_Q, we can actually measure the distance between P and Q by measuring the distance between their means.

$$|P - Q|^2 = |\mu_P - \mu_Q|^2 \tag{1}$$

where,

$$\mu_P = \frac{1}{n}\sum_{i=1}^{n}\phi(x_i)\mu_Q = \frac{1}{m}\sum_{j=1}^{m}\phi(y_j) \tag{2}$$

This ϕ function is supposed to take care of all the moments of the distribution. The ϕ function however may not be known as shown in the following proof:-

$$\begin{aligned}
&|\mu_P - \mu_Q|^2 \\
&= |\frac{1}{n}\sum_{i=1}^{n}\phi(x_i) - \frac{1}{m}\sum_{i=1}^{m}\phi(y_i)|^2 \\
&= |\frac{1}{n}\sum_{i=1}^{n}\phi(x_i) - \frac{1}{m}\sum_{j=1}^{m}\phi(y_j)|^T.|\frac{1}{n}\sum_{i=1}^{n}\phi(x_i) - \frac{1}{m}\sum_{j=1}^{m}\phi(y_j)| \\
&= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{m}\phi(x_i)^T.\phi(x_j) + \frac{1}{m^2}\sum_{i=1}^{n}\sum_{j=1}^{m}\phi(y_i)^T.\phi(y_j) - \frac{2}{mn}\sum_{j=1}^{m}\sum_{i=1}^{n}\phi(x_i)^T.\phi(y_j)
\end{aligned} \tag{3}$$

The $\phi^T\phi$ function can be replaced by a Kernel function K(x,y) that has the appropriate properties to measure the distance between 2 distributions. One of the examples of such Kernel functions are RBF Kernel Maps that map data in an infinitely long Hilbert space and match all of moments of the two distributions with some samples. Thus, we use MMD to find the distance between 2 distributions and the result is always non-negative.

1.1 Moment Matching Generative Models

Generative Models, as the name suggests, generate data. This can be done by sampling from a given distribution, let us say Gaussian, and preparing a dataset with similar properties like mean and variance. Here, the goal is to create a network that is provided with a sampled dataset from a standard distribution and expect a certain distribution as the output, e.g. the output vector should be such that when I arrange the values, the vector represents the image of a cat. Here, out objective function becomes the MMD function that basically tells us the distance or the dissimilarity between the output distribution and the wanted distribution. This model is called the moment matching generative model.

1.2 Generative Adversarial Network

Generative Adversarial Networks are extremely useful and has found many applications. Interestingly, we can think of this network as a MMGM with an MMD function as the objective function that builds an output which is similar to a distribution we are looking for. To take this judgment, we employ another MMD function that works as the classifier and, on the contrary to the MMD in the network, this MMD tries to find the difference between the generated output and the given dataset. This classifier is not pre-trained though. We train the classifier MMD and the network MMD at the same time. Thus, they have an adversarial operation where the network MMD tries to generate an output as close as possible to the desired distribution, while the classifier MMD tries to contradict it by finding the dissimilarities between the 2.

Learning.png

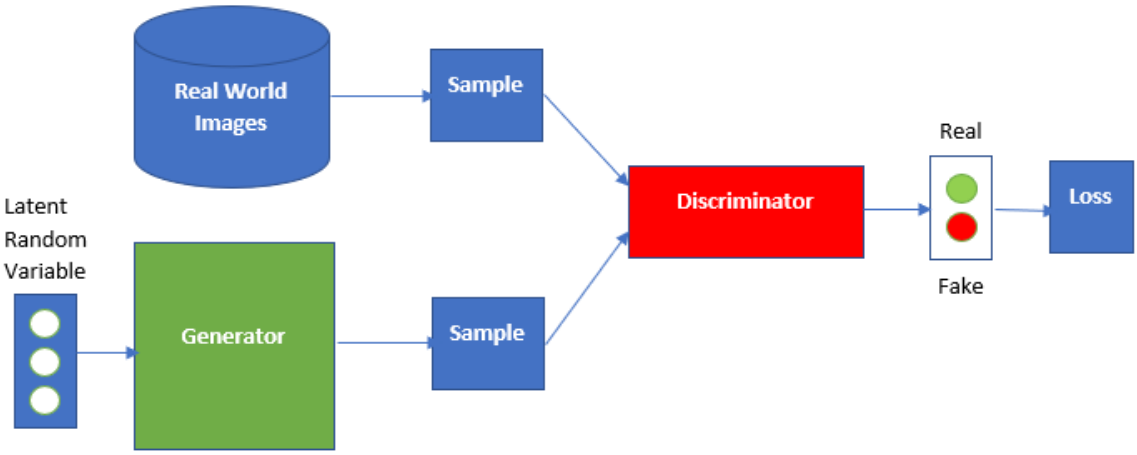

Figure 1: Adversarial Learning

1.3 Objective function

The objective function of a GAN takes care of both the minimizing and maximizing operations of the 2 MMDs. The function can be written as the following:-

$$\min_G \max_D V(D, G) \tag{4}$$

where,

$$V(D, G) = E_{x \sim P_{data}(x)}[\log D(x)] + E_{z \sim P_z(z)}[1 - \log D(G(z))] \tag{5}$$

Here $P_z(z)$ refers to the prior distribution, z is a sample from the same that is passed to the generator function G(z), whereas $x \sim P_{data}(x)$ is the real data that is fed to the discriminator network for comparison of the generated data. The discriminator then uses a sigmoid function to provide the output. So during the operation of the network, the optimization function switches from maximizing the discriminator function and then minimizing the generator function until both of them are trained and optimized.

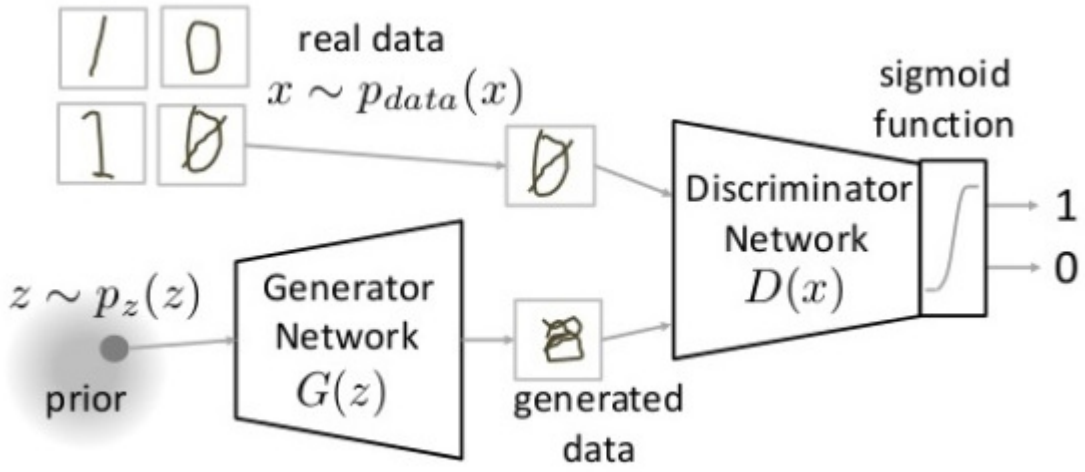

Figure 2: Training a Generative Adversarial Learning

1.4 Operation of the Optimization function

We can express the distribution generated or $P_g(x)$ as follows :-

$$x = G(z) \implies z = G^{-1}(x) \implies dz = (G^{-1})'(x)dx \implies P_g(x) = P_z(G^{-1}(x))(G^{-1})'(x) \tag{6}$$

From (5) we see that:-

$$\begin{aligned} V(D,G) &= E_{x\sim P_{data}(x)}[\log D(x)] + E_{z\sim P_z(z)}[1 - \log D(G(z))] \\ &= \int_x \mathrm{P_{data}(x)} \log \mathrm{D(x)}\, \mathrm{d}x + \int_z \mathrm{P_z(z)} \log[1 - \mathrm{D(G(z))}]\, \mathrm{d}z \\ &= \int_x \mathrm{P_{data}(x)} \log \mathrm{D(x)}\, \mathrm{d}x + \int_z \mathrm{P_z(G^{-1}(x))} \log[1 - \mathrm{D(x)}](\mathrm{G^{-1}})'\mathrm{(x)}\, \mathrm{d}x \\ &= \int_x \mathrm{P_{data}(x)} \log \mathrm{D(x)}\, \mathrm{d}x + \int_x \mathrm{P_g(x)} \log[1 - \mathrm{D(x)}]\, \mathrm{d}x \\ &= \int_x \mathrm{P_{data}(x)} \log \mathrm{D(x)} + P_g(x) \log[1 - D(x)]\, \mathrm{d}x \end{aligned} \tag{7}$$

Now, this is the expression we have to maximize wrt D given G.

$$\max_D V(D,G) = \max_D \int_x \mathrm{P_{data}(x)} \log \mathrm{D(x)} + P_g(x) \log[1 - D(x)]\, \mathrm{d}x \tag{8}$$

$$\begin{aligned} \frac{\partial(P_{data}(x) \log D(x) + P_g(x) \log[1 - D(x)])}{\partial D(x)} &= 0 \\ D^*(x) &= \frac{P_{data}(x)}{P_{data}(x) + P_g(x)} \end{aligned} \tag{9}$$

Replacing the value of $D^*(x)$ in 8 we get:-

$$\begin{aligned} & C(D,G) \\ &= \max_D V(D,G) \\ &= \max_D \int_x \mathrm{P_{data}(x)} \log \frac{\mathrm{P_{data}(x)}}{\mathrm{P_{data}(x) + P_g(x)}} + P_g(x) \log \frac{P_g(x)}{P_{data}(x) + P_g(x)}\, \mathrm{d}x \\ &= KL[P_{data}(x) || \frac{P_{data}(x)}{\frac{P_{data}(x)+P_g(x)}{2}}] + KL[P_g(x) || \frac{P_g(x)}{\frac{P_{data}(x)+P_g(x)}{2}}] - \log 4 \end{aligned} \tag{10}$$

$$\begin{aligned}
&\min_G C(G) \\
&= 0 + 0 - log4 \\
&= -log4
\end{aligned} \tag{11}$$

$$\begin{aligned}
KL[P_{data}(x)||\frac{P_{data}(x)}{\frac{P_{data}(x)+P_g(x)}{2}}] = 0 \\
when, P_{data}(x) = \frac{P_{data}(x) + P_g(x)}{2} \\
\implies P_{data}(x) = P_g(x)
\end{aligned} \tag{12}$$

In this way, the objective function finally matches the generated distribution with the wanted distribution.

2 References

- Ali Ghodsi, Lec : Deep Learning, Generative Adversarial Network, Oct 24 2017
- https://jaan.io/what-is-variational-autoencoder-vae-tutorial/