A Mathematical Introduction to GANs

Xavier Garcia

July 2, 2019

Generative adversarial networks (GANs) have achieved great success for sampling continuous data. The purpose of these notes is to give a more mathematically-minded introduction to these generative networks.

1 Setting

Suppose that we have some datapoints $x_1, ..., x_N$, where each $x_i \in \mathbb{R}^n$. Our goal is to find a way to generate more datapoints similar to the ones we have. To rigorously define this, we make two assumptions:

1. The datapoints $x_1, ..., x_N$ are actually i.i.d. samples of some unknown random variable X.
2. There exists a positive integer m and a standard multivariate Gaussian $Z \sim \mathcal{N}(0, I_m)$, where I_m denotes the $m \times m$ identity matrix, and a smooth function $G : \mathbb{R}^m \to \mathbb{R}^n$ such that $G(Z) = X$.

The first assumption reduces the problem to finding the distribution of X. The second assumption gives us some structure that we can exploit to approximate the distribution of X. These assumptions turns out to be quite flexible due to the following lemma:

Lemma 1.1. *The set of random variables*

$$\mathcal{G} := \{X = G(Z) \,|\, G : \mathbb{R}^m \to \mathbb{R}^n \text{ continuous }, Z \sim \mathcal{N}(0, I_m), m \in \mathbb{N}\}$$

is dense in the set of random variables, under the topology given by weak convergence.

Proof. The key idea is that the set of mixtures of Gaussians is dense, and the closure of $\mathcal{G}$ contains the set of mixtures of Gaussians, and hence it too must be dense. We leave the proof of this as an exercise to the reader. □

This lemma tells us that even if X doesn't satisfy assumption 2, we can find some random variable arbitrarily close to X which does. If we possess such a G as in assumption 2, we can generate new datapoints by sampling from Z and passing it through G. G is traditionally referred to as the **generator**.

The problem then reduces to finding G. To do so, we first formulate a minimization problem for which G is the answer, then we approximate this problem using the data we have. Traditionally, machine learning uses the KL divergence between $G(Z)$ and X as the objective to be minimized i.e.

$$G = \text{argmin}_G \, \text{KL}(X||G(Z)).$$

However, this objective could potentially be troublesome, since it requires knowing the densities of $G(Z)$ and X, which may not even exist. Instead, we use the Jensen-Shannon divergence i.e.

$$G = \text{argmin}_G \, \text{JS}(X||G(Z)) \tag{1}$$

where the Jensen-Shannon divergence is defined by the formula:

$$\text{JS}(X \,||\, G(Z)) = \frac{1}{2}\text{KL}\left(X \,||\, M\right) + \frac{1}{2}\text{KL}\left(G(Z) \,||\, M\right).$$

where the random variable M is the mixture distribution given by

$$M = B \cdot X + (1 - B) \cdot G(Z)$$

where B is an independent symmetric Bernoulli random variable. Since X is absolutely continuous with respect to M, we know there exists a density[1] D_X of X with respect to M. Similarly, we write $D_{G(Z)}$ for the density of $G(Z)$ with respect to M. This allows to write the KL terms explicitly. For example,

$$\text{KL}(X||M) = \mathbb{E}_{x\sim X}[\log D_X(x)]$$

and similarly for the other KL term. Moreover, we can relate $D_{G(Z)}$ and D_X due to the fact that M is a mixture of $G(Z)$ and X. More explicitly, we have the following:

Lemma 1.2. *For all x, we have the following identity:*

$$D_{G(Z)}(x) + D_X(x) = 2$$

Under stronger assumptions, we can even write out a formula for D_X:

Lemma 1.3. *If both X and $G(Z)$ had densities p_X and $p_{G(Z)}$ respectively, we can write D_X as follows:*

$$D_X(x) = \frac{2p_X(x)}{p_X(x) + p_{G(Z)}(x)}.$$

We can rewrite the GAN objective (1) in terms of D_X as follows:

$$\text{JS}(X||G(Z)) = \mathbb{E}_{x\sim X}[\log D_X(x)] + \mathbb{E}_{z\sim Z}[\log(2 - D_X(G(z)))]$$

[1] More formally, a Radon-Nikodym derivative.

where in the second term we use the fact that $D_{G(Z)} = 2 - D_X$ by Lemma 1.2. This final expression can be approximated by Monte Carlo methods[2].

Keeping Lemma 1.3 in mind, the quantity $\frac{1}{2}D_X(x)$ is viewed as the probability that a sample x was sampled from the distribution of X as opposed to $G(Z)$. Letting $D(x) = \frac{1}{2}D_X(x)$, we can write the objective in terms of D, named in the literature as the **discriminator**, as follows:

$$\mathrm{JS}(X \,||\, G(Z)) = E_{x\sim X}[\log D(x)] + \mathbb{E}_{z\sim Z}[\log(1 - D(G(z)))] + 2\log 2$$

In practice, we don't have an explicit formula for D. Instead, we make an additional assumption that D and G belong to some parametrized family of distributions i.e. there exists families of functions $\{G_\theta\}_\theta$ and $\{D_\phi\}_\phi$ as well as θ^* and ϕ^* such that $D = D_{\phi^*}$ and $G = G_{\theta^*}$. These families of functions should be flexible enough to approximate any reasonable continuous function i.e. they should be dense in the space of continuous functions. With this additional assumption, our goal has been reduced to finding θ^* and ϕ^* satisfying

$$(\theta^*, \phi^*) = \min_\theta \max_\phi L(\theta, \phi)$$

where the loss function L is given by

$$L(\theta, \phi) := E_{x\sim X}[\log D_\phi(x)] + \mathbb{E}_{z\sim Z}[\log(1 - D_\phi(G_\theta(z)))] + 2\log 2 \qquad (2)$$

Remark 1.4. *If we know the optimal (θ^*, ϕ^*), then $G_{\theta^*}(Z) = X$ and hence $D_{\phi^*}(x) = \frac{1}{2}$ for all x by Lemma 1.2. In general, the inner* max *computes the JS divergence between $G_\theta(Z)$ and X by recovering the ϕ (as a function of θ) of the discriminator, while the* min *operation recovers the θ which minimizes this divergence.*

Remark 1.5. *Even if G and D don't actually belong to these parametrized families, we can expect that G_{θ^*} and D_{ϕ^*} are "close enough" to G and D respectively, if the families are flexible enough.*

By enforcing a smooth dependence on the parameters of G_θ and D_ϕ, we can tackle the optimization problem by a double gradient descent mechanism. Namely, we start with an initial guess (θ_0, ϕ_0) and we update the nth guess (θ_n, ϕ_n) by performing the following two operations:

$$\phi_{n+1} := \phi_n + \nabla_\phi L(\theta_n, \phi_n)$$
$$\theta_{n+1} := \theta_n - \nabla_\theta L(\theta_n, \phi_n)$$

The first update attempts to minimize $-L$ (i.e. maximize L) while the second minimizes L. One could also perform updates in bulk e.g. update ϕ k times, then update θ k' times. Theoretically, it seems reasonable to set $k' = 1$ and choose $k > 1$, but this has mixed results.

[2] By this, I mean $\mathbb{E}_{x\sim X}[f(x)] \approx \frac{1}{k}\sum_{i=1}^k f(x_i)$ for samples $x_1, .., x_k$ from X.

2 Extended Discussion

2.1 Training problems

From the loss function (2), we can read off potential training concerns. For example, the only dependence on θ for L manifests through $D_\phi(G_\theta)$. In particular, this implies the following lemma:

Lemma 2.1. *The function $\theta \mapsto L(\theta, \phi^*)$ is identically equal to 0.*

Once this happens, all the gradients vanish and make the optimization strategy fail. This suggests that if we ever completely confuse the discriminator i.e. make it so that it constantly produces $\frac{1}{2}$ for all values, training automatically stops for the generator. Notice that this situation can happen early in training when the discriminator has learned much.

The other extreme is even more dangerous.

Lemma 2.2. *Suppose that some point during training, the discriminator becomes perfect i.e. $D_\phi(G_\theta(z)) = 0$ and $D_\phi(x) = 1$ for any samples $x \sim X$ and $z \sim Z$ and some parameters (θ, ϕ). Then the function $\theta \mapsto L(\theta, \phi)$ is identically equal to $2 \log 2$.*

This can happen if we train the discriminator for many steps before updating the generator. This one is more troublesome because we can't update ϕ any more, since the discriminator is perfect. Moreover, this can happen if the support of $G_\theta(Z)$ is disjoint from the support of X, which can easily occur for distributions in high dimension.