Chapter 15

Generative Adversarial Networks

Reading

1. Andrew Ng's notes on generative models
   http://cs229.stanford.edu/notes/cs229-notes2.pdf
2. The original GAN paper by Goodfellow et al. (2014)
3. "The Numerics of GANs" by Mescheder et al. (2017)

In the previous chapter, we used variational methods to build a generative model for the data. In this case, we are given samples $D = \left\{x^i\right\}_{i=1}^n$ and would like to build a model that can synthesize new data. For every data x that a decoder synthesizes at test time using latent variables z, we can calculate the likelihood

$$x \sim p_v(x|z), \text{ for any } z \sim N(0, I).$$

This likelihood is an indicator of how unlikely the data x is under z. Models for which we can calculate such likelihood are called explicit generative models, i.e., they give a sample x and also report its likelihood. In this chapter, we will look an alternative class of generative models that are implicit, i.e., they only give a sample x but do not report its likelihood.

A Generative Adversarial Network (GAN) consists of two neural networks: a Generator and a Discriminator. The Generator works in the same way as the decoder in a variational auto-encoder. Given a sample z from some distribution, most commonly a standard normal, we train a neural network to generate a sample

$$x = g_v(z).$$

GANs differ from explicit models in how they train the generator, the discriminator is used for this purpose. We will look at this next.

15.1 Two-sample tests and Discriminators

We will first take a short trip into an area of statistics known as decision theory. Consider two datasets coming from two distributions $p(x)$ and $q(x)$

$$D_1 = \left\{x^1, \ldots, x^n, :\ x^k \sim p(x)\right\}$$
$$D_2 = \left\{x^1, \ldots, x^n, :\ x^k \sim q(x)\right\}.$$

We would like to check if these two distributions are the same given access to only their respective datasets D_1 and D_2. Let us define the *null hypothesis* which claims that the two distributions are the same.

⚠ The concept of a hypothesis here is different from what we saw in generalization/VC-theory. Hypothesis in decision theory simply means our hunch about a particular situation, e.g., $p = q$.

$$H_0 : p = q$$

The alternate hypothesis is

$$H_1 : p \neq q.$$

The goal of the so-called "two-sample test" is to decide whether H_0 is true or not. A typical two-sample test will construct a statistic (recall from Chapter 7 that a statistic is any function of the data)

$$\hat{t}$$

out of the two datasets, e.g., their individual means, their variances, and will use this statistic to *accept or reject* the null hypothesis, i.e., decide whether H_0 is true or false.

Let's say that we pick a threshold t_α, and the test statistic $\hat{t}$ is the difference of the means

$$\hat{t} = \left| \frac{1}{n} \sum_{x \in D_1} x - \frac{1}{n} \sum_{x \in D_2} x \right|.$$

Level of a test A statistician will then say that the null hypothesis is valid with *level* α if

$$\mathbb{P}_{D_1 \sim p,\ D_2 \sim p} \left(\hat{t} > t_\alpha\right) \leq \alpha. \tag{15.1}$$

In other words, if the null hypothesis were true (both D_1 and D_2 are drawn from the same distribution p) and if the probability of our empirical statistic $\hat{t}$ being larger than some *chosen* threshold t_α is smaller than some *chosen* probability α, then we know that the two distributions are the same despite only having finite data to check. The threshold α is called the p-value in the statistics literature and you will have seen statements like "gene marker XX is correlated with disease YY with p-value of 10^{-3}" or "smokers and non-smokers have different distributions of cancers with p-value of 10^{-3}".

Power of a test The power of a two-sample test is the probability of rejecting the null hypothesis when it is actually false. We want tests with a large power, i.e., we like

$$\mathbb{P}_{D_1 \sim p, D_2 \sim q} \left(\hat{t} > t_\alpha\right) \tag{15.2}$$

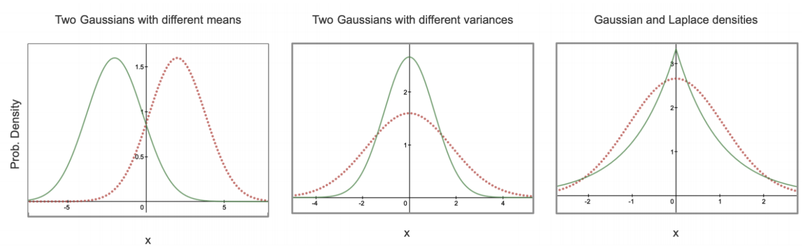
being large if the two datasets D_1 and D_2 are drawn from two different distributions p and q respectively.

The key point to remember about two-sample tests is that they let us check if two distributions are the same without knowing anything about the distributions. We only need access to the samples and can run this test. This is fundamentally different than say

$$\mathrm{KL}(q \,||\, p) = \int q(x) \log \frac{q(x)}{p(x)} \,\mathrm{d}x$$

where we need to know the probabilities $q(x), p(x)$ to compute the distance between distributions.

Example 15.1. A two-sample test requires three things, a statistic $\hat{t}$, a level α and a threshold for the statistic t_α. The latter two are numbers that a statistician can pick, e.g., picking $\alpha = 0.05$ is an accepted standard in most biological studies.



15.2 Building the Discriminator in a GAN

Finding two-sample test statistics for arbitrary distributions is difficult, especially for high-dimensional problems where the samples are natural images. The key idea behind a Generator Adversarial Network (GAN) is to learn the statistic $\hat{t}$.

A good statistic is the one that lets us distinguish between data that comes from Nature's distribution and data that is synthesized by our generative model. This statistic, which is called the discriminator in GAN, is a critic of the generative model's results. It has a *high power* in (15.2) if the generated samples are different from those of Nature. Why? Because in this case for most thresholds t_α that we can pick, the power of the two-sample test in (15.2) will be large.

The discriminator should also be sound, i.e., if the two distributions are indeed the same (e.g., if our generator is as good as good as Nature's renderer), the discriminator should have a *low level* α in (15.1).

We are going to train a binary classifier

$$d_u : \mathcal{X} \mapsto [0, 1]$$

that will act as the discriminator in a GAN. You should think of the decision boundary of this binary classifier as the difference of the test statistic and our threshold $\hat{t} - t_\alpha$.

We next create a dataset to train this classifier. Given n images from Nature's distribution $p(x)$ and the distribution of our generator's images $q(x)$, we will label the former with $y = 1$ and the latter with $y = 0$ to create a joint dataset:

$$\begin{aligned} D_1 &= \left\{(x^i, 1)_{i=1,\dots,n} : \ x^i \sim p(x)\right\} \\ D_2 &= \left\{(x^i, 0)_{i=1,\dots,n} : \ x^i \sim q(x)\right\} \\ D &= D_1 \cup D_2. \end{aligned}$$

Fitting d_u on this problem can be done simply using the logistic loss wherein d_u is modeling the log-odds

$$\log \frac{\mathbb{P}(y = 1|x)}{\mathbb{P}(y = 0|x)} = d_u(x).$$

The logistic loss is therefore

$$u^* = \underset{u}{\text{argmin}} -\frac{1}{n} \sum_{x \sim D_1} \log d_u(x) - \frac{1}{n} \sum_{x \sim D_2} \log(1 - d_u(x)). \tag{15.3}$$

Observe that this is the same logistic loss that we are used to; the only difference being that the entire dataset has $2n$ samples with all the ones in D_1 having labels $y = 1$ and all the ones in D_2 having labels $y = 0$.

What is the ideal discriminator? The population risk corresponding to the discriminator's objective in (15.3) is

$$d^* = \underset{d}{\text{argmax}} \underset{x \sim p}{\mathbb{E}} [\log d(x)] + \underset{x \sim q}{\mathbb{E}} [\log(1 - d(x))]. \tag{15.4}$$

We can take the variational derivative of this objective (just like you did in HW 3 to compute the optimal classifier in the bias-variance tradeoff) to get

$$d^*(x) = \frac{p(x)}{p(x) + q(x)}. \tag{15.5}$$

Observe that the ideal discriminator is $1/2$ if the two distributions p and q are the same. The intuitive reason for this is that if the data D were really coming from the same distribution, we would never be able to fit a logistic classifier to get better than 50% error because D_1 and D_2 have different labels in spite of having similar input data.

Think of you would use our discriminator to build a two-sample test for a given dataset. If given two datasets D_1 and D_2 labeled as above

$$\hat{t} := \frac{1}{n} \sum_{x \in D_1} \mathbf{1}_{\{d_u(x) > 0\}} + \frac{1}{n} \sum_{x \in D_2} \mathbf{1}_{\{d_u(x) < 0\}}$$

and the threshold $t_\alpha = 1/2$. This construction is an example of what is called a "classifier-based two-sample test"; you can read more about it at Lopez-Paz and Oquab (2016).

It can be shown that if the two distributions are not the same, the

⚠ Notice how rigorous theory is used as an inspiration for developing GANs. This is a common theme that you will see in the deep learning literature; the models may seem *ad hoc* and sprung out of sheer intuition, but the reason they work well is often because there are sound theoretical principles behind them. Building this skill requires studying the classical curriculum (ML, statistics, optimization) but being creative in applying this curriculum with deep networks.

⚠ For a functional

$$L[d] = \int \log d(x) p(x) \, \mathrm{d}x$$

the variational derivative is

$$\frac{\delta L}{\delta d}(x) = \frac{p(x)}{d(x)}.$$

Similarly, the variational derivative for

$$L[d] = \int \log(1 - d(x)) q(x) \, \mathrm{d}x$$

is

$$\frac{\delta L}{\delta d}(x) = \frac{q(x)}{1 - d(x)}.$$

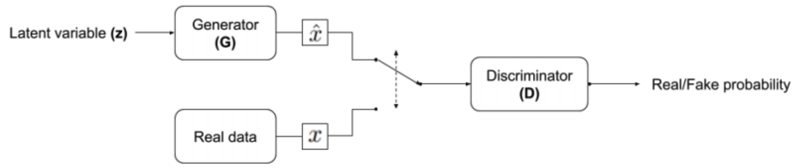


Figure 15.1: Schematic of the architecture in a GAN

power of the two-sample test is an increasing function of the statistic $\hat{t}$. Therefore if we wanted to maximize the power, maximizing the test statistic $\hat{t}$ of the discriminator is a good idea. This makes the discriminator more and more sensitive to the differences between samples from p and q.

15.3 Building the Generator of a GAN

The second key idea in a GAN is that the generator

$$g_v : \mathcal{Z} \to \mathcal{X}$$

that maps the latent space $\mathcal{Z} \subset \mathbb{R}^m$ to data space $\mathcal{X}$ is trained to *minimize* the power of the two-sample test.

The generator g_v wants to synthesize data that look like they came from Nature's distribution $p(x)$. As the generator's distribution q comes closer to p, the accuracy of the discriminator d_u will degrade (it cannot distinguish between them as easily) and thereby discriminator will be forced to make its test statistic more sensitive to subtle differences between the two distributions.

15.4 Putting the discriminator and generator together

The GAN objective combines two objectives: the discriminator updates its weights u to maximize the power and the generator updates its weights v to minimize the power. We will write the population version of the optimization problem as follows.

$$\min_v \max_u \; E_{x \sim p(x)} \; [\log d_u(x)] + E_{x \sim q(x)} \; [\log(1 - d_u(x))] \qquad (15.6)$$

Let us fill in a few more details. The dataset of real images consists of samples from Nature's distribution $p(x)$, so we will write it as a finite sum over our dataset $D = \left\{x^i \sim p\right\}_{i=1}^n$. The generator uses samples z from some generic distribution, e.g., a standard Gaussian distribution.

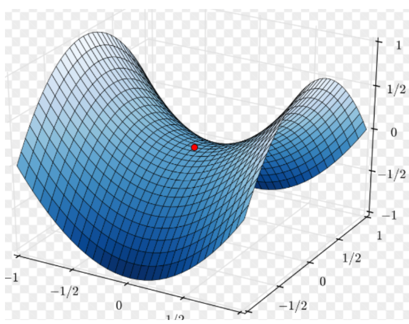
$$\min_v \max_u \frac{1}{n} \sum_{x \in D} \; [\log d_u(x)] + E_{z \sim N(0,I)} \left[\log(1 - d_u(g_v(z)))\right]. \qquad (15.7)$$

Training a GAN The objective in (15.7) is an example of a min-max optimization problem. Such problems are quite difficult to solve and this is why training GANs is quite difficult. In practice, we typically resort to a few crude tricks. We sample a mini-batch of real images $\left\{x^1, \ldots, x^b\right\}$ and another mini-batch of noise vectors $\left\{z^1, \ldots, z^b\right\}$. Using these two mini-batches

1. we update the generator g_v using the gradient of the objective with respect to v.
2. update the discriminator d_u using the gradient of the loss with respect to u.

There is no need for the Reparametrization Trick here because there is no expectation being taken over parametrized distributions. This is a big benefit of the GAN formulation as compared to variational inference; the former does not have to be careful while picking a variational family and complex deep networks can be used as the generator or the discriminator easily. Let us next make a few comments about the objective in (15.7).

Solving min-max problems is difficult This is a min-max problem: the generator is minimizing the objective and the discriminator is maximizing the objective. Such problems are hard to solve in optimization especially with gradient descent techniques. Consider an example of a saddle point



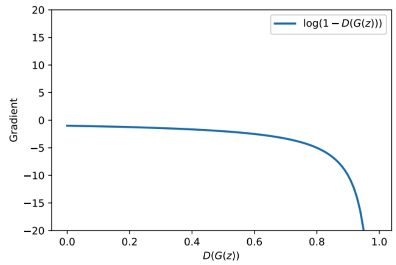
where the loss function increases in one direction and decreases in the other direction. Finding the solution of the min-max objective involves finding the saddle point. It is easy to appreciate that depending on how many steps of gradient descent we take for either of the min/max players we risk falling down or climbing up the hill. There are many many other other factors that make solving such problems hard, e.g., learning rate, momentum, stochastic gradients if we are using mini-batches. Hyper-parameters are very tricky to pick while training GANs and this is often called "instability of training".

A harsh discriminator inhibits the training of the generator The generator has a much more difficult task than the discriminator. During early stages of training, the generator needs to learn how to synthesize images whereas the discriminator can easily distinguish between bad images generated by the generator and good ones from our original dataset using very similar test statistics, e.g., an average of the RGB values all the pixels.

The gradient of the second term in the objective is

$$\nabla_v \log(1 - d_u(g_v(z))) = -\frac{\nabla_v \, d_u(g_v(z))}{1 - d_u(g_v(z))}.$$

As a function of $d_u(g_v(z))$ the second term in the objective thus looks like

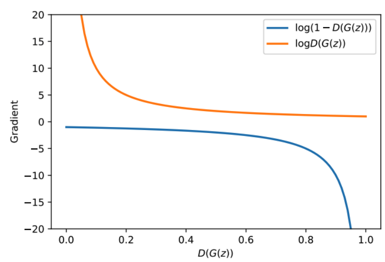


In other words, the gradient with respect to the generator's weights v is essentially zero if the generator is not working well (this is the case when $d_u(g_v(z))$ predicts a large negative value). This does not allow the generator to learn well; it is essentially like your advisor shooting down all your ideas.

Most GAN implementations therefore modify the second term in the objective to be

$$-\mathop{\mathbb{E}}_{z\sim N(0,I)}\left[\log d_u(g_v(z))\right]$$

which does not suffer from the small gradient problem.



Synthesizing new images from a GAN The generator samples latent factors $z \sim N(0, I)$ at test time to synthesize new images. The discriminator is not used at test time.

15.5 How to perform validation for a GAN?

For variational generative models, we can use the log-likelihood of synthesized images to obtain some understanding of whether the model is working well. If the log-likelihood of new images is similar to the log-likelihood of images in the training data then the new images are good at least as far as the model is concerned even if they may have perceptual artifacts.

Doing so is not so easy for implicit models because they do not output the likelihood of the generated samples. Run the generator a few times to eyeball the quality of images it generates

But this is a very heuristic and qualitative metric.

Frechet Inception Distance (FID) A number of other metrics exist for evaluating generative models. One popular one is the so-called Frechet Inception Distance (FID) where we take any pre-trained model for classification, in this case people typically use the Inception architecture, and evaluate

$$\text{FID}(p, q) = \|\mu_p - \mu_q\|_2^2 + \text{trace}\left(\Sigma_p + \Sigma_q - 2\left(\Sigma_p \Sigma_q\right)^{1/2}\right).$$

where μ_p, Σ_p are the mean and covariace of the features of an Inception network when real images are fed to it and similarly μ_q, Σ_q are the mean/-covariance of the features when GAN-generated images are fed to the same network.

The above formula is the Wasserstein distance between the two densities p, q, There are many similar techniques such as the Maximum Mean Discrepency (MMD) that can be used to understand the discrepancy between the two distributions once the features are computed using some pre-trained model on their respective images.

Roughly speaking, the evaluation methodology in generative models, especially for images, is quite flawed. This is not a new phenomenon in machine learning/statistics because it is a intrinsically difficult problem to measure when two distributions are the same given only finite data from them. The problem is exacerbated in deep generative models because deep networks are very good at over-fitting, e.g., GANs can often end up memorizing the training data, they can generate very realistic images that are essentially the same as those in the training data. Nevertheless, a lot of techniques exist to make GANs synthesize high-quality images. See some examples at Brock et al. (2018); Karras et al. (2017).

The key behind the empirical success of GANs is to convert a problem about estimating distributions, sampling from them etc. into a classification problem. Deep networks are extremely good at classification as compared to other problems like regression, reconstruction etc. and GANs leverage this remarkably. This is a trick that you will do well to remember when you use deep networks in the future: typically you will always get better results if you manage to rewrite your problem as a classification problem. I suspect the real reason for this is that we do not have good regularization techniques for deep networks for non-classification problems.

15.6 The zoo of GANs

Due to the numerous issues with GANs, there have been a large number of variants and attempts to improve their empirical performance. They fall mainly into the following categories.

1. Optimization tricks to train the generator-discriminator pair in a more stable fashion.
2. New loss functions that change the binary cross-entropy loss of the discriminator to something else. A majority of papers, including the example we saw above, fall into this category.
3. Characterizing the kind of critical points, equilibria of the training process; this is a similar line of analysis as the study of the energy landscape of deep networks for standard supervised learning.
4. Connections with variational inference suggest that GANs and their training techniques are essentially variational inference in disguise (Nowozin et al., 2016).
5. Coming up with new ways of evaluating generative models.

In addition to the above lines, there are many other novel and interesting applications such as Cycle-GANs and conditional-GANs.

Bibliography

Brock, A., Donahue, J., and Simonyan, K. (2018). Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.

Lopez-Paz, D. and Oquab, M. (2016). Revisiting classifier two-sample tests. *arXiv preprint arXiv:1610.06545*.

Mescheder, L., Nowozin, S., and Geiger, A. (2017). The numerics of gans. In *Advances in Neural Information Processing Systems*, pages 1825–1835.

Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pages 271–279.