Generative Adversarial Networks

Mohammad Aminian

January 14, 2022

Introduction to GANs

Generative adversarial networks, or GANs (Goodfellow et al., 2014c), are another generative modeling approach based on differentiable generator networks. Generative adversarial networks are based on a game theoretic scenario in which the generator network must compete against an adversary. The generator network directly produces samples $x = g(z; \theta^{(g)})$. Its adversary, **the discriminator network**, attempts to distinguish between samples drawn from the training data and samples drawn from the generator. The discriminator emits a probability value given by $d(x; \theta^{(d)})$, indicating the probability that $\mathbf{x}$ is a real training example rather than a fake sample drawn from the model. The simplest way to formulate learning in generative adversarial networks is as a zero-sum game, in which a function $v(\theta^{(g)}, \theta^{(d)})$determines the payoff of the discriminator. The generator receives $-v(\theta^{(g)}, \theta^{(d)})$ as its own payoff. During learning, each player attempts to maximize its own payoff, so that at convergence

$$g^* = arg_g minmax_d v(g, d). \tag{1}$$

The default choice for v is

$$v(\theta^{(g)}, \theta^{(d)}) = \mathbb{E}_{x \sim pdata} log d(x) + \mathbb{E}_{x \sim pmodel} log(1 - d(x)). \tag{2}$$

This drives the discriminator to attempt to learn to correctly classify samples as real or fake. Simultaneously, the generator attempts to fool the classifier into believing its samples are real. At convergence, the generator's samples are indistinguishable from real data, and the discriminator outputs $\frac{1}{2}$ everywhere. The discriminator may then be discarded. The main motivation for the design of GANs is that the learning process requires neither approximate inference nor approximation of a partition function gradient.

When $max_d v(g, d)$ is convex in $\theta^{(g)}$ (such as the case where optimization is performed directly in the space of probability density functions), the procedure is guaranteed to converge and is asymptotically consistent.

Unfortunately, learning in GANs can be difficult in practice when g and d are represented by neural networks and $max_d v(g, d)$ is not convex. Goodfellow (2014) identified nonconvergence as an issue that may cause GANs to underfit. In general, simultaneous gradient descent on two players' costs is not guaranteed to reach an equilibrium. Consider, for example, the value function $v(a, b) = ab$, where one player controls a and incurs cost ab, while the other player controls b and receives a cost $-ab$. If we model each player as making infinitesimally small gradient steps, each player reducing their own cost at the expense of the other player, then a and b go into a stable, circular orbit, rather than arriving at the equilibrium point at the origin. Note that the equilibria for a minimax game are not local minima of v. Instead, they are points that are simultaneously minima for both players' costs. This means that they are saddle points of v that are local minima with respect to the first player's parameters and local maxima with respect to the second player's parameters. It is possible for the two players to take turns increasing then decreasing v forever, rather than landing exactly on the saddle point, where neither player is capable of reducing its cost. It is not known to what extent this nonconvergence problem affects GANs.

Goodfellow (2014) identified an alternative formulation of the payoffs, in which the game is no longer zero-sum, that has the same expected gradient as maximum likelihood learning whenever the discriminator is optimal. Because maximum likelihood training converges, this reformulation of the GAN game should also converge, given enough samples.

Unfortunately, this alternative formulation does not seem to improve convergence in practice, possibly because of suboptimality of the discriminator or high variance around the expected gradient.

In realistic experiments, the best-performing formulation of the GAN game is a different formulation that is neither zero-sum nor equivalent to maximum likelihood, introduced by Goodfellow et al. (2014c) with a heuristic motivation. In this best-performing formulation, the generator aims to increase the log-probability that the discriminator makes a mistake, rather

than aiming to decrease the log- probability that the discriminator makes the correct prediction. This reformulation is motivated solely by the observation that it causes the derivative of the generator's cost function with respect to the discriminator's logits to remain large even in the situation when the discriminator confidently rejects all generator samples.

Stabilization of GAN learning remains an open problem. Fortunately, GAN learning performs well when the model architecture and hyperparameters are care- fully selected. Radford et al. (2015) crafted a deep convolutional GAN (DCGAN) that performs very well for image synthesis tasks, and showed that its latent rep- resentation space captures important factors of variation.

The GAN learning problem can also be simplified by breaking the generation process into many levels of detail. It is possible to train conditional GANs (Mirza and Osindero, 2014) that learn to sample from a distribution $p(x|y)$ rather than simply sampling from a marginal distribution $p(x)$. Denton et al. (2015) showed that a series of conditional GANs can be trained to first generate a very low-resolution version of an image, then incrementally add details to the image. This technique is called the LAPGAN model, due to the use of a Laplacian pyramid to generate the images containing varying levels of detail. LAPGAN generators are able to fool not only discriminator networks but also human observers, with experimental subjects identifying up to 40 percent of the outputs of the network as being real data.

One unusual capability of the GAN training procedure is that it can fit proba- bility distributions that assign zero probability to the training points. Rather than maximizing the log-probability of specific points, the generator net learns to trace out a manifold whose points resemble training points in some way. Somewhat para- doxically, this means that the model may assign a log-likelihood of negative infinity to the test set, while still representing a manifold that a human observer judges to capture the essence of the generation task. This is not clearly an advantage or a disadvantage, and one may also guarantee that the generator network assigns nonzero probability to all points simply by making the last layer of the generator network add Gaussian noise to all the generated values. Generator networks that add Gaussian noise in this manner sample from the same distribution that one obtains by using the generator network to parametrize the mean

of a conditional Gaussian distribution.

Dropout seems to be important in the discriminator network. In particular, units should be stochastically dropped while computing the gradient for the generator network to follow. Following the gradient of the deterministic version of the discriminator with its weights divided by two does not seem to be as effective. Likewise, never using dropout seems to yield poor results.

While the GAN framework is designed for differentiable generator networks, similar principles can be used to train other kinds of models. For example, **self- supervised boosting** can be used to train an RBM generator to fool a logistic regression discriminator (Welling et al., 2002).

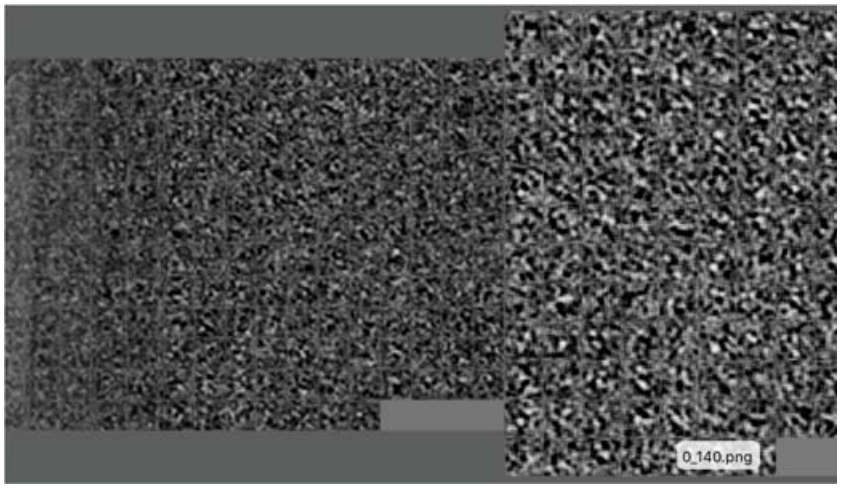
The discriminator network. When modeling images, the discriminator network is typi cally a standard CNN. Using a secondary neural network as the discriminator net work allows the GAN to train both networks in parallel in an unsupervised fashion. These discriminator networks take images as input, and then output a classification. The gradient of the output of the discriminator network with respect to the synthetic input data indicates how to make small changes to the synthetic data to make it more realistic.

The generative network. The generative network in GANs generates data (or images) with a special kind of layer called a deconvolutional layer (read more about deconvo lutional networks and layers in the following sidebar). During training, we use backpropagation on both networks to update the generating network's parameters to generate more realistic output images. The goal here is to update the generating network's parameters to the point at which the discriminating network is sufficiently "fooled" by the generating network because the output is so realistic as compared to the training data's real images.

Some cool GAN applications

We have established that the generator learns how to forge data. This means that it learns how to create new synthetic data which is created by the network and looks real and created by humans. Before going into details about the GAN code, I'd like to share the results of a recent paper (code is available online https://github.com/hanzhanggit/StackGAN) in which a GAN was used to synthesize forged images starting with a text description. The results were impressive. The first column is the real image in the test set and all the other columns are the images generated from the same text description in Stage-I and Stage-II of StackGAN.

Now let us see how a GAN can learn to forge the MNIST dataset. In this case, it is a combination of GAN and ConvNets used for the generator and the discriminator networks. In the beginning, the generator creates nothing understandable, but after a few iterations, synthetic forged numbers are progressively clearer and clearer. In the following figure, the panels are ordered by increasing training epochs and you can see the quality improvement among the panels:



The improved image is as follows:

One of the coolest uses of GAN is doing arithmetics on faces in the generator's vector Z. In other words, if we stay in the space of synthetic forged images, it is possible to see things like this: [smiling woman] - [neutral woman] + [neutral man] = [smiling man], or like this: [man with glasses] - [man without glasses] + [woman without glasses] = [woman with glasses].

Bibliography

[1] TensorFlow 1.x Deep Learning Cookbook: Over 90 unique recipes to solve artificial-intelligence driven problems with Python(Antonio Gulli, Amita Kapoor)

[2] Deep Learning Adaptive Computation and Machine Learning series(Ian Goodfellow,Yoshua Bengio and Aaron Courville)

[3] Deep Learning A Practitioner's Approach(Josh Patterson and Adam Gibson)