Article from

Predictive Analytics and Futurism

April 2018
Issue 17

The Forgery Game: Generative Adversarial Networks

By Michael Niemerg

Imagine a not-too-distant future. You open your mailbox to find a pretty ordinary-seeming catalog. You start to flip through it. Inside, you find pictures of beautiful, smiling people. You see perfectly manicured lawns and perfect bedrooms. The catch: None of this is real. These images weren't even created using computer graphics. All these images were created by a model—by a generative adversarial network (GAN).[1,2] Don't believe this is possible? There are already images of fake people that look eerily realistic[3] and ways to manipulate an image to turn that smile into a frown.[4]

What is a generative adversarial network? How does it create synthetic images of people and things that are nearly indistinguishable from real photos? The first thing we need to do is parse the moniker itself. The "generative" part of generative adversarial networks refers to what the model is doing: generating synthetic data. The "adversarial" refers to how it is trained—in an adversarial fashion between two competing models. The "networks" refer to the model form, which are neural networks (while there is no requirement that generative adversarial models *must* be neural networks, this is the primary focus of active research in the area).

TRAINING GANS

Let's dive a little more into how these models are trained. GANs are created via two competing networks: a generator that creates synthetic data and a discriminator whose job it is to distinguish the real data from the synthetic data. This adversarial connection is the whole key to the process. By putting the models in competition, the generator is forced to successively get better at creating data that looks real while the discriminator gets increasingly more adept at separating real data from synthetic data. A common analogy used to describe GANs is to think of the generator model as an artwork forger, trying to pass off forgeries as the real thing, while the discriminator plays the role of the curator trying to identify the real art and reject the forgeries. The forger gets continually better at generating the fake artwork but the curator also improves at spotting the real art apart from the forgeries.

GAN models are neural networks. While the relationship between the generator and the discriminator is unique, all the typical rules and structure of training neural networks apply to both. If the jargon of neural networks is foreign to you, simply remember that a neural network is a predictive model. It will take in some data, have parameters that will be fit by optimizing an objective, and ultimately produce output (the synthetic data for the generator, and the probability of data being real or synthetic for the discriminator).

Now let's get a little more precise on the algorithm for GANs.

GAN ALGORITHM

For each round of training:

- Generate random points from latent space (a good choice would be random numbers from a normal distribution) and create the synthetic data by feeding the random points into the generator.
- Combine this synthetic data with the real data.
- Train the discriminator to distinguish between these real and synthetic values.
- Update the generator to fool the discriminator:
  - Freeze the discriminator so that its weights do not change.

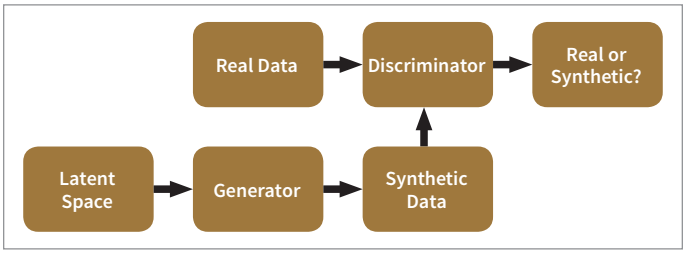
- Feed the generator random points from latent space as input.

- The generator will convert these random points to synthetic data.

- The frozen discriminator will then classify this synthetic data as "real" or "synthetic."

- Update the weights in the generator to alter how it creates its synthetic data so that it can more easily fool the discriminator.

Figure 1
A Representation of the GAN Model-Building Process



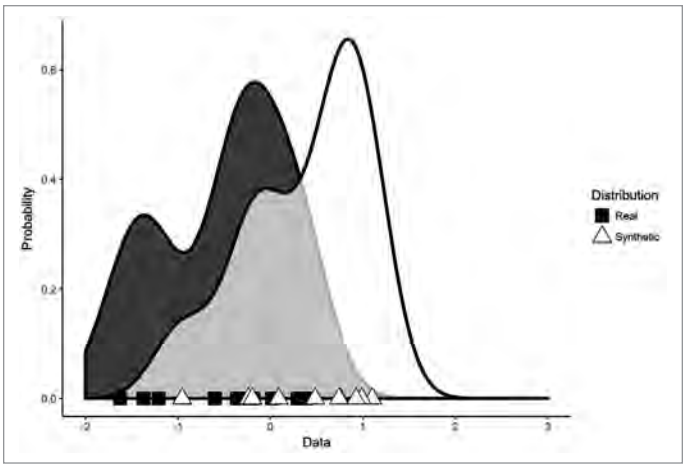
The last step above can seem a bit curious so let's look more closely at what is happening. In more precise terms, this step in the process is trying to minimize the difference between two vectors of numbers (with each entry in the vectors corresponding to an observation). Keeping in mind that the discriminator is being fed a series of synthetic observations, the first vector is the discriminator's prediction of whether each of these observations is real or synthetic. The second is simply a vector of targets that say that each observation is real. Because the generator is trying to fool the discriminator, it wants to get them as close as possible. However, while training with this objective, the generator is unable to manipulate the discriminator directly (in fact, being frozen, the discriminator doesn't change at all in this last step) but the generator is still able to indirectly alter the first vector (the discriminator's predictions) by altering its own weights so that its generated output becomes harder for the discriminator to distinguish from the actual data.

Ultimately, the generator is doing a good job when the discriminator can't tell the difference between synthetic data and real data (e.g., the predicted probability of either is 50 percent). The coolest part? Throughout this whole training process, the generator has no access to the real images! It learns to create them without ever having direct access to them.

Another way to think about what the model is doing is to think about our real sample data as coming from a high-dimensional, data-generating distribution. When training a GAN, our training set is really a sample of data points from this data-generating distribution. The GAN model uses this sample data to learn about the structure of the entire data-generating distribution so that it can learn how to approximate new samples from it.

For an illustrative example, see Figure 2. Our data set to build our GAN is a sample of points (black boxes) from the data-generating distribution (gray distribution). Our model learns an (imperfect) representation of that distribution (white distribution) from which we can draw samples (white triangles).

Figure 2
Data-Generating Distribution and GAN Approximation



CHALLENGES WITH TRAINING GANS

Currently, generative adversarial modeling is still an active area of research. There are several ways in which GANs can fail or in which training them can produce fickle results.

The most common is simply instability in training. For instance, training the model with the same parameters might work well in one training run only to produce poor results in another run without any changes to the model parameters other than different random number initializations.

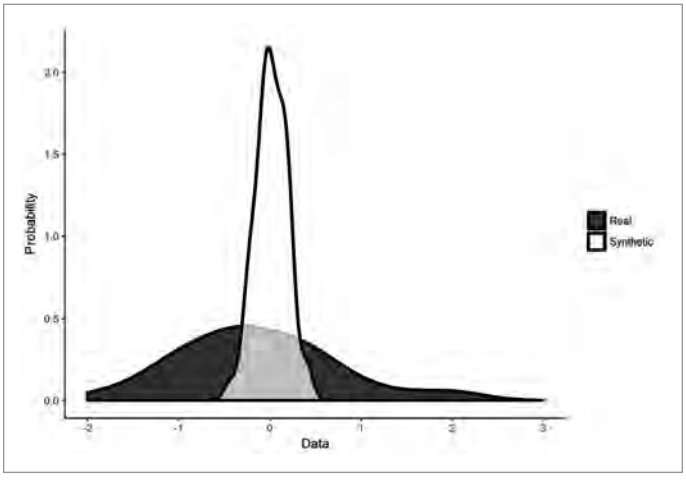
Another problem with GANs is that measuring the quality of the synthetic data can be difficult. While both the generator and the discriminator have a loss function, these loss functions are really only optimizing the competition against its adversary. In a regression problem, we know that higher R-squared is better

and, in a classification problem, that higher accuracy is better (*ceteris paribus*). If our task is generating realistic synthetic images, however, our real objective is independent of the nominal value of the loss function but is instead tied to how convincing the image is to a human. Because it is hard to come up with a good loss function for how different the synthetic picture of a bedroom is from a real bedroom, it can be hard to tell exactly when one GAN model performs better by simply checking metrics. We need to actually examine our sample output.

Another difficulty with training GANs is that they have a tendency to collapse into similar output for different input from the latent space. Part of the reason for this is that the GAN model can only look at each instance in isolation when determining whether a data point is real or synthetic. Why is this problematic? Well, imagine, for instance, that you wanted some synthetic data representing the rolls of a six-sided die. If I presented you with a 0 or a 7 you would easily recognize those data points as unrealistic. However, what if I presented you with a 4? That seems to be a very plausible die roll. What if I then generated for you a never-ending series of 4s as synthetic data? If you are constrained to only being able to look at one data point at a time to judge whether an instance looks real (i.e., we are memoryless like a GAN), you can't discriminate these obviously synthetic data points from real points. This is problematic. We need some way of relating observations to each other to tell the difference.

In Figure 3, we can see an example of a degenerative GAN. The GAN fails to learn a good representation of the true data-generating distribution, instead only learning to reproduce frequent values that lie near the mean of the data-generating distribution.

Figure 3
Data-Generating Distribution and GAN Approximation: Degenerative Example



Another challenge with GANs is one that faces all predictive models: They inherit the biases of the data used to train them. Say, for instance, we are training a model to generate images of bedrooms. Let's also suppose only a small percentage of bedrooms contain yellow bedsheets and that none of these bedrooms make it into our training set for the GAN model. What could likely happen is that our model will not learn to associate yellow bedsheets with bedrooms and our synthetic images will contain no yellow bedsheets even though they exist in the real world. Our model can only reconstruct the data-generating distribution to the extent that it is faithfully represented in our training data.

PRACTICAL TIPS AND ADVANCED ARCHITECTURES

Multiple techniques exist for aiding the training of GANs. Some techniques include: modifications to the loss function used in training, incorporating common neural network regularization techniques into the training phase, and adding some extra challenge to the discriminator by introducing noise to its input. Many of these techniques are incorporated into advancements to the original GAN algorithm.

A few of the advanced GAN algorithms are particularly noteworthy. Deep convolutional GANs (DCGANs)[4] improve upon GANs by offering refinements to the architecture of the neural networks used to train them. Wasserstein GANs[5] add several wrinkles to GANs, including using a loss function whose numerical value corresponds more closely with the true quality of the synthetic data. Furthermore, the idea of mini-batch discrimination[6] was created to counter the tendency of models to collapse to a narrow output range by adding distance information about other examples from within each training mini-batch to the discriminator.

Generally, research on GANs is proceeding at a rapid clip. In all likelihood, significant improvements have been made to GANs between when I wrote this article and the time it went to print.

DOES IT MATTER TO ACTUARIES?

Much of the work with GANs to date has been on synthetic image and audio generation but that is quickly changing. Will GANs ever make their way to the insurance or health care sectors? The future is still to be seen, but the potential is there as the quality of the algorithms mature and the use cases become more apparent.

One possible use for GANs could be to generate data synthetically to feed into other predictive models when training data is scarce. Various types of data set augmentation are already common practice when creating neural networks for image analysis. GANs could simply become another extension of this practice.

Another more creative use for GANs could be in the realm of data-sharing. Imagine being able to share the data needed to

build predictive models without sharing the data itself. Instead of training the predictive model with real data, one party could train a GAN on its data to "encrypt" it. The other party could then generate synthetic data from the GAN and use that synthetic data to actually train the ultimate predictive model. The only thing that needs to be shared is the neural network itself. In this way, data insight could be shared without actually sharing data.

These use cases are speculative at the moment but not unrealistic. It's still too early to tell whether GANs rise to prominence as another commonplace method in the modeler's toolbox or whether they remain a curiosity. ■

Michael Niemerg, FSA, MAAA, is an actuary at Milliman in Chicago. He can be reached at *michael.niemerg@milliman.com.*

ENDNOTES

1 Goodfellow, Ian J., et al. 2014. Generative Adversarial Nets. Advances in Neural Information Processing Systems, *https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf* (accessed Feb. 9, 2018).

2 Creswell, Antonia, et al. Generative Adversarial Networks: An Overview. Cornell University Library, Oct. 19, 2017, *https://arxiv.org/abs/1710.07035* (accessed Feb. 9, 2018).

3 Vincent, James. All of These Faces Are Fake Celebrities Spawned by AI. *The Verge*, Oct. 30, 2017, *https://www.theverge.com/2017/10/30/16569402/ai-generate-fake-faces-celebs-nvidia-gan* (accessed Feb. 9, 2018).

4 Radford, Alec., Luke Metz, and Soumith Chintala. Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks. Cornell University Library, Nov. 19, 2015, *https://arxiv.org/abs/1511.06434v2* (accessed Feb. 9, 2018).

5 Arjovsky, Martin, Soumith Chintala, and Léon Bottou. Wasserstein GAN. Cornell University Library, Jan. 26, 2017, *https://arxiv.org/abs/1701.07875* (accessed Feb. 9, 2018).

6 Salimans, Tim, et al. Improved Techniques for Training GANs. Cornell University Library, June 10, 2016, *https://arxiv.org/abs/1606.03498* (accessed Feb. 9, 2018).