Loss Functions

GANs try to replicate a probability distribution. They should therefore use loss functions that reflect the distance between the distribution of the data generated by the GAN and the distribution of the real data.

How do you capture the difference between two distributions in GAN loss functions? This question is an area of active research, and many approaches have been proposed. We'll address two common GAN loss functions here, both of which are implemented in TF-GAN:

- **minimax loss**: The loss function used in the paper that introduced GANs (https://arxiv.org/abs/1406.2661).
- **Wasserstein loss**: The default loss function for TF-GAN Estimators. First described in a 2017 paper (https://arxiv.org/abs/1701.07875).

TF-GAN implements many other loss functions as well.

One Loss Function or Two?

A GAN can have two loss functions: one for generator training and one for discriminator training. How can two loss functions work together to reflect a distance measure between probability distributions?

In the loss schemes we'll look at here, the generator and discriminator losses derive from a single measure of distance between probability distributions. In both of these schemes, however, the generator can only affect one term in the distance measure: the term that reflects the distribution of the fake data. So during generator training we drop the other term, which reflects the distribution of the real data.

The generator and discriminator losses look different in the end, even though they derive from a single formula.

Minimax Loss

In the paper that introduced GANs, the generator tries to minimize the following function while the discriminator tries to maximize it:

$$E_x[log(D(x))] + E_z[log(1 - D(G(z)))]$$

In this function:

- `D(x)` is the discriminator's estimate of the probability that real data instance x is real.
- E_x is the expected value over all real data instances.
- `G(z)` is the generator's output when given noise z.
- `D(G(z))` is the discriminator's estimate of the probability that a fake instance is real.
- E_z is the expected value over all random inputs to the generator (in effect, the expected value over all generated fake instances G(z)).
- The formula derives from the cross-entropy (/machine-learning/glossary#cross-entropy) between the real and generated distributions.

The generator can't directly affect the `log(D(x))` term in the function, so, for the generator, minimizing the loss is equivalent to minimizing `log(1 - D(G(z)))`.

In TF-GAN, see minimax_discriminator_loss and minimax_generator_loss (https://github.com/tensorflow/gan/blob/master/tensorflow_gan/python/losses/losses_impl.py) for an implementation of this loss function.

Modified Minimax Loss

The original GAN paper notes that the above minimax loss function can cause the GAN to get stuck in the early stages of GAN training when the discriminator's job is very easy. The paper therefore suggests modifying the generator loss so that the generator tries to maximize `log D(G(z))`.

In TF-GAN, see modified_generator_loss (https://github.com/tensorflow/tensorflow/blob/2007e1ba474030fcce840b0b8a599558e7d5998f/tensorflow/contrib/gan/python/losses/python/losses_impl.py) for an implementation of this modification.

Wasserstein Loss

By default, TF-GAN uses Wasserstein loss (https://arxiv.org/abs/1701.07875).

This loss function depends on a modification of the GAN scheme (called "Wasserstein GAN" or "WGAN") in which the discriminator does not actually classify instances. For each instance it outputs a number. This number does not have to be less than one or greater

than 0, so we can't use 0.5 as a threshold to decide whether an instance is real or fake. Discriminator training just tries to make the output bigger for real instances than for fake instances.

Because it can't really discriminate between real and fake, the WGAN discriminator is actually called a "critic" instead of a "discriminator". This distinction has theoretical importance, but for practical purposes we can treat it as an acknowledgement that the inputs to the loss functions don't have to be probabilities.

The loss functions themselves are deceptively simple:

Critic Loss: `D(x) - D(G(z))`

The discriminator tries to maximize this function. In other words, it tries to maximize the difference between its output on real instances and its output on fake instances.

Generator Loss: `D(G(z))`

The generator tries to maximize this function. In other words, It tries to maximize the discriminator's output for its fake instances.

In these functions:

- `D(x)` is the critic's output for a real instance.
- `G(z)` is the generator's output when given noise z.
- `D(G(z))` is the critic's output for a fake instance.
- The output of critic D does *not* have to be between 1 and 0.
- The formulas derive from the earth mover distance (https://wikipedia.org/wiki/Earth_mover%27s_distance) between the real and generated distributions.

In TF-GAN, see wasserstein_generator_loss and wasserstein_discriminator_loss (https://github.com/tensorflow/gan/blob/master/tensorflow_gan/python/losses/losses_impl.py) for implementations.

Requirements

The theoretical justification for the Wasserstein GAN (or WGAN) requires that the weights throughout the GAN be clipped so that they remain within a constrained range.

Benefits

Wasserstein GANs are less vulnerable to getting stuck than minimax-based GANs, and avoid problems with vanishing gradients. The earth mover distance also has the advantage of being a true metric: a measure of distance in a space of probability distributions. Cross-entropy is not a metric in this sense.

Previous
← GAN Training (/machine-learning/gan/training)

Next
Check Your Understanding (/machine-learning/gan/check) →