Thoughts on the role of the generator in Bayesian GANs

Pavel Izmailov

September 27, 2017

1 Introduction

In this short note I formulate some thoughts regarding the role of the generator in Bayesian GANs and understanding the model in general.

In the original Bayesian GAN paper it was proposed to use Bayesian neural networks for the generator and discriinator, and a new formulation of the GAN problem was provided. In this formulation we have the following conditional distributions for the parameters of generator and discriminator

$$p(\theta_g|\theta_d, z) \propto \prod_{i=1}^{n_g} D(G(z^{(i)}, \theta_g), \theta_d) p(\theta_g),$$

$$p(\theta_d|\theta_g, z, X) \propto \prod_{i=1}^{n_d} D(x^{(i)}, \theta_d) \prod_{i=1}^{n_g} (1 - D(G(z^{(i)}, \theta_g), \theta_d)) \cdot p(\theta_d).$$

In the experiments the model showed very good results in terms of semi-supervised accuracy and sample diversity. I think it would be very exciting and important to understand the model better and to see why it works so well. In all experiments the discriminator network was actually deterministic and was trained in a standard way, so probably what makes bayesian GANs work so well is the generator.

The distribution that is used for the generator weights is not completely straightforward. If the discriminator is fixed, then we are trying to force the samples from the generator to be distributed according to the density induced by the discriminator with some complicated prior. This leads us to the question of why do we need a generator? Why can't we just sample directly from this distribution?

2 Bayesian GAN without a generator

I propose the following simplified formulation which is tightly related to Bayesian GANs.

$$p(z|D) \propto \frac{D(z)}{1 - D(z)} p(z),$$

$$L(\theta_d|Z, X) \propto \prod_{i=1}^{n_d} D(x^{(i)}, \theta_d) \prod_{i=1}^{n_g} (1 - D(z^{(i)}, \theta_g), \theta_d)).$$

The discriminator can be Bayesian or deterministic. For simplicity I will suppose it's deterministic, because it worked well with Bayesian GANs. Here I also changed the likelihood $D(z)$ to $D(z)/(1 - D(z))$ for the discriminator for the following reason. Suppose we have access to the true data distribution and prior distribution of z, and train an optimal classifier D_{opt}. Then

$$D_{opt}(x) = \frac{p_{data}(x)}{p_{data}(x) + p_{prior}(x)} \Rightarrow p_{data}(x) = \frac{D_{opt}(x) p_{prior}(x)}{1 - D_{opt}(x)}.$$

This simplified formulation has some nice properties. In particular with infinite data and infinite samples from prior, the generator would learn to sample from the data distribution. However, I don't think that this statement is very useful, as it doesn't give ant insights about how the model will behave in practice with finite datasets and finite capacity models.

Another good propperty is that in this formulation it is clear that we will not get mode-collapse. Indeed, the generator is always sampling from the distribution that is induced by the discriminator. If our sampling will work ok, the only way the mode-collapse can occur is if the discriminator collapses to just outputing high probabilities for a small number of data points, which doesn't make sense.

Finally, in this formulation we don't need to specify a generator architecture, which might limit the samples' complexity in standard GANs. The relation between this formulation and Bayesian GANs might explain why the former produces good samples and semi-supervised accuracy, despite using a simple architecture for the generator.

3 Do we need a generator?

I think the role of the generator in Bayesian GANs is not completely clear. It would be interesting to find out that we can get rid of it, or to understand why we can't. Even if we conclude that the discriminator is indeed useful, we will probably have more understanding of the features that we want it to have, how to propperly select it's architecture and so on.

One possible way of how the generator can help is through inducing a more informative distribution over samples. I believe however that the prior over the samples that is induced by the generator can't be very informative. Indeed, let's look at how samples are generated.

$$G(z, \theta_g) = b_H + net_{H-1}(z, \theta_g),$$

where b_H is the bias on the last layer of the generator network. In the current implementation we put a broad prior over the parameters of the network, meaning that the bias variable can approximate any reasonable vector in the data space with fairly high density (is this true?). Even if the distribution of the $net_{H-1}(z, \theta_g)$ is highly structured and provides much more support for reasonable images, we would loose this support, because we whiten this distribution by convolving it with a non-informative b_H.

The intuitive explanation of what I tried to articulate in the previous paragraph is that for any two sample vectors with not very high it should be almost equally probable that sampling a random network we would generate one or the other, because we can transform the network generating the first sample to the network generating the second sample by just changing the bias on the last layer, for which the distribution is fairly broad.

Another way in which a generator could help is that we are using the same sample of weights to produce multiple samples in the data space. This can provide additional regulariztion. The ability to provide multiple samples might also lead to faster exploration of the data distribution, becuse we can simultaneously produce highly varying samples. However I'm not sure if this happens in practice. Looking at the samples that I got with Bayesian GANs on SVHN, I would say that in almost all cases I just see two or three different pictures with different distortions, but these two pictures differ dramatically between different samples of the generator weights.