Metropolitan Generative Adversarial Networks

Anonymous Author(s)
Affiliation
Address
email

1 Introduction

Generative adversarial networks (GANs) presented a radically new way to do density estimation: They only implicitly represent the density of the data via a classifier that distinguishes real from generated data. Traditionally, density estimation has been done with a model that can easily compute the likelihood of the observed data.

GANs iterate between updating a discriminator D and a generator G, where G generates new (synthetic) samples of data, and D attempts to distinguish samples of G from the real data. Typically, in this setup, D is thrown away at the end of training, and only G is kept for generating new synthetic data points. In this work we propose the Metropolitan GAN (MGAN) that constructs a new generator G' that "wraps" G using the information contained in D.

The MGAN uses Markov chain Monte Carlo (MCMC) methods to sample from the distribution implicitly defined by the discriminator D. This is based on the notion that the discriminator classifies between the generator G and a real data distribution:

$$D(\mathbf{x}) = \frac{p_D(\mathbf{x})}{p_D(\mathbf{x}) + p_G(\mathbf{x})}\,, \tag{1}$$

where p_G is the (intractable) density of samples from the generator G, and p_D is the real data density implied by the discriminator D. If GAN training reaches its global equilibrium then this discriminator distribution is equal to the true distribution on the data ($p_D = p_R$) [Goodfellow et al., 2014].

We use an MCMC *independence sampler* to get samples from p_D using multiple samples from G (as the proposal). As a corollary: Given a perfect discriminator D and a decent (but imperfect) generator G, we can obtain exact samples from the true data distribution p_R. Standard MCMC implementations cannot achieve this because MCMC requires (unnormalized) densities for the target p_D and the proposal p_G. Neither of these quantities are available in GANs. However, MCMC can be performed using only the ratio p_D/p_G, which is implied by the discriminator D.

We can also avoid the burn-in issues that usually plague MCMC methods. Recall that via the detailed balance property, if the marginal distribution on a Markov chain at time step k matches the target p_D then the marginal at time step $k+1$ will also follow p_D. Typically, it is not possible to get an initial sample from the target distribution because MCMC is typically applied to generate samples of *parameters* from a Bayesian posterior distribution. We have no example samples of parameters from these distributions to start from. However, in MGANs, we are sampling from the *data* distribution. Therefore, by initializing the chain at a sample of real data, we are already initializing it from the correct distribution.

We also derive a simple deterministic approximation to MCMC that is very accurate in the regimes that GANs operate in. An important aspect of using the discriminator in MCMC is to have well calibrated probabilities in D. Typical GAN usage does not directly use D and can survive poorly calibrated discriminators, however these probabilities become crucial for accept/reject probabilities. We use a held out set recalibration routine for the MGAN setup; and show that indeed the raw probabilities from D in typical GAN training are poorly calibrated.

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

2 Background

In this section we quickly review the notation and equations with MCMC and GANs.

2.1 MCMC Methods

MCMC methods attempt to draw a chain of samples $\mathbf{x}_{1:K} \in \mathcal{X}^K$ marginally from a target distribution $\mathbf{x} \sim p^\star$. The Markov chain in MCMC works by drawing a first sample from an initial distribution $\mathbf{x}_0 \sim p_0$. In general, the proposed points $\mathbf{x}' \in \mathcal{X}$ are drawn from a conditional distribution $\mathbf{x}' \sim q(\mathbf{x}'|\mathbf{x}_k)$ (the proposal distribution). However, in this work we are using an independence sampler, which means $\mathbf{x}' \sim q(\mathbf{x}')$. The proposal $\mathbf{x}'$ is accepted with probability:

$$\alpha(\mathbf{x}', \mathbf{x}_k) = \min\left(1, \frac{p^\star(\mathbf{x}')q(\mathbf{x}_k|\mathbf{x}')}{p^\star(\mathbf{x}_k)q(\mathbf{x}'|\mathbf{x}_k)}\right) \in [0, 1]\,, \tag{2}$$

which for an independence sampler is:

$$\alpha(\mathbf{x}', \mathbf{x}_k) = \min\left(1, \frac{p^\star(\mathbf{x}')q(\mathbf{x}_k)}{p^\star(\mathbf{x}_k)q(\mathbf{x}')}\right)\,. \tag{3}$$

If the point $\mathbf{x}'$ is accepted then $\mathbf{x}_{k+1} = \mathbf{x}'$, otherwise $\mathbf{x}_{k+1} = \mathbf{x}_k$. This acceptance probability is known as the Metropolis-Hastings rule from which we derive the name Metropolitan GAN. Note that when estimating the distribution on $p^\star$ one must include the duplicates that are a result of rejections in $\mathbf{x}'$.

The detailed balance condition implies that if $\mathbf{x}_k \sim p^\star$ exactly then $\mathbf{x}_{k+1} \sim p^\star$ exactly as well. Additionally, even if $\mathbf{x}_k$ is not exactly distributed according to $p^\star$, its KL divergence to $p^\star$ will always decrease as k increases [Murray and Salakhutdinov, 2008, Cover and Thomas, 2012].

In our setup, in order to obtain independent samples, we sample $\mathbf{x} \sim p_0$ and then run the chain a fixed number of iterations K to obtain the final sample produced by the "wrapped GAN" G'. That is, $\mathbf{x}_K$ is the output of G'. Independent chains are used for multiple samples from G'. Note that the number of samples K is meant to be ancillary to the state of the Markov chain $\mathbf{x}_k$, using the state of the Markov chain to determine when to stop has the potential to introduce bias [Cowles et al., 1999].

2.2 GANs

GANs implicitly model the distribution on data $\mathbf{x} \sim p_R$ by sampling synthetic data points from a generator; we use p_R to refer to the unknown true distribution on the data $\mathbf{x}$. The generator works by taking a sample from a Gaussian $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then transforming it with a generator network $G \in \mathbb{R}^L \to \mathcal{X}$: $\mathbf{x} = G(\mathbf{z})$, $\mathbf{z} \in \mathbb{R}^L$. This implies a distribution on the data $\mathbf{x} \sim p_G$. However, because G is often a complex and non-invertible function, p_G is generally intractable to compute.

The discriminator function $D \in \mathcal{X} \to [0, 1]$ is merely a standard binary classifier, that outputs the probability a data point is real (as opposed to being sampled from p_G). The training of a GAN interleaves updates where D is trained as a standard classifier to maximize its performance classifying real samples from generated samples; and G is trained to minimize the performance of D. Therefore, GAN training forms a game between D and G. The original work of Goodfellow et al. [2014] showed that if this training game converges to its global equilibrium, then

$$p_G = p_R, \quad D = \frac{p_R}{p_R + p_G} = \frac{1}{2}\,. \tag{4}$$

In practice, it is often found that it is much easier to get D close to its Bayes optimal solution $(p_R/(p_R + p_G))$ than G close to the true distribution p_R. This motivates our approach of wrapping an imperfect G to obtain an improved G' using a near perfect D.

2.3 Related Work

There have been a few other works that in some way combine GAN training and MCMC methods. The work of Song et al. [2017] uses a GAN-like training procedure to improve the proposal used in MCMC to sample from an externally provided target density $p^\star$. They use a RealNVP network [Dinh

et al., 2016] as the proposal as invertibility is important for computation of the acceptance probability α. The difference with this work is best summarized as: Song et al. [2017] uses GANs to accelerate MCMC whereas we use MCMC to enhance the solution from a GAN. A similar approach to Song et al. [2017] was taken in Kempinska and Shawe-Taylor [2017], but with respect to finding improved proposals for particle filters as opposed to MCMC.

The GAN approach to density estimation is complementary to the lesser known technique of *density ratio estimation* [Sugiyama et al., 2012]. Here, the generator G is typically fixed, or simple, and the density of the data is determined entirely by combining Bayes' rule and the learned classifier D. The MGAN, in effect, combines elements of both approaches to construct the wrapped GAN G'.

3 Methods

In this section we show how to sample from the distribution p_D implied by the discriminator D. If we assume that D is an optimal discriminator between the generator p_G and *some* alternative distribution p_D, then:

$$D = \frac{p_D}{p_D + p_G} \implies \frac{p_D}{p_G} = \frac{1}{D^{-1} - 1}\,, \tag{5}$$

and if D is perfect then $p_D = p_R$. We can use an MCMC independence sampler with a target distribution of $p^\star = p_D$ and a proposal distribution of p_G. This gives an acceptance probability of:

$$\alpha(\mathbf{x}', \mathbf{x}_k) = \min\left(1, \frac{p_D(x')}{p_G(\mathbf{x}')}\frac{p_G(\mathbf{x}_k)}{p_D(\mathbf{x}_k)}\right) = \min\left(1, \frac{D(\mathbf{x}_k)^{-1} - 1}{D(\mathbf{x}')^{-1} - 1}\right)\,, \tag{6}$$

which is computable using only the discriminator D and no densities.

Calibration A key element to this is *calibration*: The probabilities for D must not merely provide a good AUC score, but be on the right scale. Put in other terms, if one were to warp the probabilities of the perfect discriminator in (5) it may still suffice for standard GAN training, but it will not work in the MCMC procedure defined in (6). To test the calibration of the discriminator, we use a calibration test statistic defined in Dawid [1997] on a *held out* data set of samples $\mathbf{x}_{1:N}$ with real/fake labels $y_{1:N} \in \{0, 1\}^N$:

$$Z := \frac{\sum_{i=1}^N y_i - D(\mathbf{x}_i)}{\sqrt{\sum_{i=1}^N D(\mathbf{x}_i)(1 - D(\mathbf{x}_i))}} \in \mathbb{R}\,. \tag{7}$$

If D is well calibrated, i.e., y is indistinguishable from a $y \sim D(\mathbf{x})$, then $Z \sim \mathcal{N}(0, 1)$. This means that large magnitude values for Z, i.e., ($|Z| > 2$), reject the hypothesis that D is well-calibrated. To correct any miscalibration we then use the held out calibration set to calibrate D using either logistic regression, isotonic regression, or beta calibration [Kull et al., 2017].

Initialization The other key aspect in any MCMC chain procedure is how one initializes the first sample $\mathbf{x}_0$. In this process we can initialize $\mathbf{x}_0$ with a random draw of real data (ideally held out). Now, marginally $\mathbf{x}_0 \sim p_R$ exactly, and if D is perfect, then by detailed balance $\mathbf{x}_k \sim p_R$ exactly as well, even if $p_G \neq p_R$. However, if the generator G is too poor we will never receive a proposal sample $\mathbf{x}'$ good enough to accept and replace the real data sample $\mathbf{x}_0$ with a generated one. Another interesting note is that we do not actually need to store real data samples for this initialization procedure. Since (6) is merely a function of the discriminator scores, one must merely take a sample from the distribution on $D(\mathbf{x})$ where $\mathbf{x} \sim p_R$. This score $D(\mathbf{x}_0)$ forms a "bar raiser" that $D(\mathbf{x}')$ must come close to, but not necessarily exceed, to be accepted.

Note that in this context the assumption that D is perfect can be weakened for two reasons: 1) Because we recalibrate the discriminator, the decision boundaries must be correct, but the probabilities can be suboptimal. 2) Because the discriminator is only ever evaluated at samples from G or the initial real sample $\mathbf{x}_0$, D only needs to be accurate on the manifold of samples from G and the real data. This is almost as equivalent to saying that the discriminator must rank samples from G (and real samples) the same way that the exact discriminator would.

Some strategy must be employed if we do not accept a single synthetic sample after we have exhausted the computational budget for a single chain K. A simple approach is to simply replay the same chain

using the same samples, but instead initializing at a generated sample. This, of course, invalidates the guarantee that $\mathbf{x}_K \sim p_R$. However, the potential discrepancy is observable by monitoring the portion of chains that reach time step K without accepting a single generated sample.

Deterministic approximation It is also possible to replace the Markov chain with a deterministic approximation that is more parallelizable. In Figure **??** we show the acceptance probabilities for different levels of the current state $D(\mathbf{x}_k)$ and the proposal $D(\mathbf{x}')$. The qualitative behavior of the curves are easily summarized. In all curves, if $D(\mathbf{x}') \geq D(\mathbf{x}_k)$ then $\mathbf{x}'$ is always accepted. As $D(\mathbf{x}_k) \to 0$ the region where $D(\mathbf{x}') < D(\mathbf{x}_k)$ becomes a linear ramp; and as $D(\mathbf{x}_k) \to 1$ it becomes a step function. If p_G and p_R are well separated (which is believed to often be the case in GANs [Arjovsky et al., 2017]), and D is near perfect, then $D(\mathbf{x}_0) \approx 1$: The acceptance probability becomes a step function if $\mathbf{x}_0 \sim p_R$. Therefore, $D(\mathbf{x}_k)$ will stay near one and the acceptance function will be step function like. In this scenario, we can approximate, $\alpha(\mathbf{x}', \mathbf{x}_k) \approx \mathbb{I}\{D(\mathbf{x}') > D(\mathbf{x}_k)\}$, and therefore G' takes the sample in the chain with the maximum score D. However, this is with the provision that we beat the score of a randomly chosen sample of real data $D(\mathbf{x})$. This maximum procedure is a reduction and is therefore easily parallelizable.

4 Results

We use an illustrative toy examples and real GANs on real data to illustrate the improvements made by the wrapped classifier G'.

4.1 Toy Data

We first demonstrate the MGAN on the toy two Gaussians example presented in Arjovsky et al. [2017]. Figure **??** shows how the convergence of samples to the true distribution over MCMC iterations. Note that if we don't strictly require all chains to end with a synthetic sample then the MGAN samples match the true distribution exactly at any number of MCMC iterations K.

4.2 Real Data

For real data experiments we considered three different data sets: CelebA [Liu et al., 2015], CIFAR-10 [Torralba et al., 2008], and LSUN [Yu et al., 2015]. We consider the traditional DC-GAN [Radford et al., 2015] and the popular WGAN [Arjovsky et al., 2017]. We first examine the performance of the discriminator on held out data with and without calibration for DC-GAN. For WGAN we can only use the calibrated discriminator as it does not output a probability. We also show the results of the calibration statistic Z from (7). These results confirm our expectation that the raw discriminators do not output well calibrated probabilities.

To evaluate the performance of the resulting wrapper generator G', we plot inception scores per epoch. The error bars are computed using a t-test on the variation per batch across different splits of the inception score. We see a consistent and statistically significant boost in inception score from G' above G.

In Figure **??** we also plot the inception score, per MCMC iteration, at epoch XXX. We see that most of gains are made in the first XXX iterations, but smaller gains continue up to iteration XXX.

We also consider the distribution on discriminator scores. In Figure **??**, the distributions on scores from real and fake images are well separated, but still contain *some overlap*. We can see that MCMC shifts the distribution of the fakes to match the distribution on true images. In this case, our deterministic approximation focuses too much on the mode as a result of the overlap, but still makes an improvement over the base generator G.

Finally, in Figure **??** we show some qualitative examples of samples comparing the base generator G and the improved MCMC version G'.

5 Discussion

The wrapped GAN G' will indeed have K times the computational burden of G when drawing samples. However, the large computational burden of GANs is during training not during drawing

samples (test). Therefore, in applications where the test speed is not the bottleneck, the wrapped GAN G' is a good option.

The deterministic approximation often has very similar behavior and performance to the full MGAN. This validates the common view that base of support between p_G and p_R have little overlap. However, applications where there is non-negligible overlap between p_G and p_R are perhaps the most interesting because proper calibration of the acceptance probabilities α are the most critical in these cases.

Along these lines, we have shown the raw discriminator in GANs are poorly calibrated. This is not a large surprise, but to our knowledge is the first time research has evaluated the discriminator in this way.

A direction for possible future extensions to this work include finding appropriate stopping rules for the MCMC chain that introduce minimal bias. In our setup, we must run the chain until at least receiving a single accept. However, ending the chain at the first accept can potentially introduce a bias into the resulting distribution.

6 Conclusions

We have presented a way to incorporate the knowledge embedded in the discriminator D into an improved generator G' in a principled manner using MCMC methods. Our method is based on the premise that the discriminator D is essentially "ahead" of the generator G. The principled MCMC setup stochastically selects among samples from G to correct biases in the generator G. This requires some modification to the MCMC equations as the neither the densities p_G nor p_R are available. This method can perform the final coordinate descent move to optimize G assuming D is near perfect. Further improvements for fancier proposals can potentially make these results possible with even shorter chains K.

References

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of the International Conference on Machine Learning*, volume 70, pages 214–223, 2017.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

M. K. Cowles, G. O. Roberts, and J. S. Rosenthal. Possible biases induced by MCMC convergence diagnostics. *Journal of Statistical Computation and Simulation*, 64(1):87–104, 1999.

A. P. Dawid. Prequential analysis. *Encyclopedia of Statistical Sciences*, 1:464–470, 1997.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of Advances in Neural Information Processing Systems*, pages 2672–2680. Curran Associates, Inc., 2014.

K. Kempinska and J. Shawe-Taylor. Adversarial sequential Monte Carlo. In *Bayesian Deep Learning (NIPS Workshop)*, 2017.

M. Kull, T. S. Filho, and P. Flach. Beta calibration: A well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 54, pages 623–631, 2017.

Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3730–3738, 2015.

I. Murray and R. Salakhutdinov. Notes on the KL-divergence between a Markov chain and its equilibrium distribution. 2008.

A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

J. Song, S. Zhao, and S. Ermon. A-NICE-MC: Adversarial training for MCMC. In *Proceedings of Advances in Neural Information Processing Systems*, pages 5140–5150. Curran Associates, Inc., 2017.

M. Sugiyama, T. Suzuki, and T. Kanamori. *Density Ratio Estimation in Machine Learning*. Cambridge University Press, 2012.

A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30 (11):1958–1970, 2008.

F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.