Reweighted Generative Adversarial Networks

R Devon Hjelm Athul Paul Jacob Tong Che Sergey M Plis Yoshua Bengio

February 19, 2017

Abstract

TODO

1 Introduction

Generative models provide a means of representing the underlying structure in data. While the objective is to faithfully generate data with representative statistics, the strength of generative models lies partially in representing the data as a hypothetical compressed version of its underlying structure. Generative adversarial networks [GANs, 6] are a unique generative learning framework that use two separate models with opposing, competing, or *adversarial* objectives. In contrast to models which generate the data from a parameteric distribution, such as directed graphical models (e.g., Helmholtz machines [2], variational autoencoders [VAEs, 10], etc) or undirected graphical models (e.g., restricted Boltzmann machines [RBMs, 8], deep Boltzmann machines [DBMs, 13]), GANs do not rely on maximum likelihood estimation [MLE, 3] to train. Rather, training GANs requires only back-propogating a learning signal originating from a loss function defined on the discriminator.

This framework is powerful, as it trains a generator without relying on MLE, which necessitates some degree of uncertainty on the generated images to work. The consequence of this is that models trained with MLE-based algorithms will generate samples that lack real-world qualities (i.e., sharpness in images [5]). For continuously valued data, GANs have been shown to be able to generate diverse and realistic samples from high-dimensional large-scale data [12], which qualitatively can be more representative of data samples than other methods.

However, GANs face a serious limitation as the data needs to be generated from a continuous function that is completely differentiable. For discrete data, generation necessitates a non-continuous function, such as a step function, so that a back-propogated learning signal originating from the discriminator cannot be computed exactly. This situation is analogous to that in training Helmholtz machines with discrete units (i.e., sigmoid belief networks [SBNs, 14]), where advanced learning algorithms are necessary to train the inference network [11, 1], while with continuous variables re-parameterization works [10]. A re-parameterization approximation exists for discrete units in the form of the Gumbel softmax, an approach known as the "Gumbel trick" [7, 9]. However, this approach is limited, as it only works with simple directed models with single layers of discrete latent variables, and it has been yet have been show to work with GANs.

In this work, we introduce a novel learning objective based on a proposed target distribution that converges to the hypothetical true distribution of the data as the discriminator is optimized. The objective can be used to define a KL-divergence loss, which can be used to train the generator on discrete data. This objective can also be re-interpreted as an alternative loss on the discriminator output, which can be used to train the generator using backprop with continuous-valued data data. We demonstrate this loss on a variety of discrete and continuous datasets, the former of which we show fails using a variety of re-parameterization tricks for back-propagation, including the Gumbel softmax. Our work shows that the generator function need not be trained to "fool" the discriminator: it only needs to be trained to set the generated samples at the decision boundary of the discriminator.

2 Methods

2.1 Generative Adversarial Networks

Generative adversarial networks [GANs, 6] are a two-model generative framework where a generator model is pitted against a discriminatory adversary. The generator is composed of a prior distribution, $p(\mathbf{z})$, and a generator function, $G(\mathbf{z})$ which maps from the space of $\mathbf{z}$ to the space of $\mathbf{x}$. The objective of the discriminator is to correctly distinguish between data and samples from the generator model by maximizing:

$$\mathbb{E}\left[\log D(\mathbf{x})\right]_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})} + \mathbb{E}\left[\log(1 - D(G(\mathbf{z}))\right]_{\mathbf{z}\sim p(\mathbf{z})}, \tag{1}$$

where $D(.)$ is a discriminator function with output in $[0, 1]$. The generator is trained to "fool" the generator, that is minimize $\mathbb{E}\left[\log(1 - D(G(\mathbf{z}))\right]_{\mathbf{z}\sim p(\mathbf{z})}$. Together, two models play a minimax game with value function:

$$\min_G \max_D V(G, D) = \mathbb{E}[\log D(\mathbf{x})]_{\mathbf{x}\sim p(\mathbf{x})} + \mathbb{E}[\log(1 - D(G(\mathbf{z})))]_{\mathbf{z}\sim p(\mathbf{z})}. \tag{2}$$

Samples from the generator are not drawn from a distribution with parameters determined by the generator network, but the are the raw output from $G(.; \boldsymbol{\psi})$, so that the generative objective involves one continuous function with parameters from both models: $D(G(.; \boldsymbol{\psi}); \boldsymbol{\phi})$. This allows for training a generative model with a relatively simple fitness function and back-propagation, rather than a complex likelihood function, and which does not suffer from some of the learning difficulties as MLE.

In practice, the generator, rather than being trained to minimize the above value function, can be trained to maximize $\log(D(G(\mathbf{z})))$ or $\log(D(G(\mathbf{z}))) - \log(1 - D(G(\mathbf{z})))$, which can alleviate some learning issues related to the discriminator loss saturating early in learning and improve stability. With basic GANs, both generator and discriminator are feed forward networks, $D(.; \boldsymbol{\phi})$ and $G(.; \boldsymbol{\psi})$, while in deep convolutional GANs [DCGAN, 12], the discriminator and generator are convolutional neural networks with strided and fractionally-strided convolutional layers, respectively.

In order for a generating function, $G(.; \boldsymbol{\psi})$, with real-valued parameters, $\boldsymbol{\psi}$, to generate discrete-valued outputs, $\mathbf{x}$, $G(.; \boldsymbol{\psi})$ cannot be fully continuous. A natural choice for the generator would be a continuous function, $\mathbf{y} = f(\mathbf{z})$ (e.g., a feed forward network), followed by a step function at 0.5 so that:

$$x_i = \phi(y_i) = \begin{cases} 1, & \text{if } y_i \geq 0.5 \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where $\mathbf{x} = \{x_i\}$ and $\mathbf{y} = \{y_i\}$. However, as the gradients that would otherwise be used to train the generator do not naturally back-propagate through discontinuous functions, it is not possible to train the generator from the discriminator error signal. Approximations for the back-propagation signal exist, which we address in more detail in the Results section. In order to address the issues more conclusively, we introduce a proposal target distribution for the generator distribution.

2.2 Target distribution for the generator

First consider the optimum discriminator function, $D(\mathbf{x})^\star$, from the discriminator objective given a fixed generator $G(.)$ [6]:

$$D^\star_G(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})}, \tag{4}$$

where $p_{\text{data}}(\mathbf{x})$ is the data distribution, and $p_g(\mathbf{x})$ is the distribution as generated by $p(\mathbf{z})$ and $G(\mathbf{z})$. Given this optimal discriminator, the density of the data can then be written as:

$$p_{\text{data}}(\mathbf{x}) = p_g(\mathbf{x}) \frac{D^\star_G(\mathbf{x})}{1 - D^\star_G(\mathbf{x})}. \tag{5}$$

Which indicates that, in the limit of a perfect discriminator, the true distribution can be perfectly estimated from said discriminator and any fixed generator. However, it is unlikely that such a perfect discriminator is learnable or even exists. One can then posit the following estimator for the true distribution, given an imperfect discriminator, $D(\mathbf{x})$:

$$\tilde{p}(\mathbf{x}) = \frac{1}{Z} p_g(\mathbf{x}) \frac{D(\mathbf{x})}{1 - D(\mathbf{x})}, \tag{6}$$

where the marginal term, $Z = \sum_{\mathbf{x}} g_g(\mathbf{x}) \frac{D(\mathbf{x})}{1 - D(\mathbf{x})}$, guarantees that $\tilde{p}(\mathbf{x})$ is a proper probability distribution. Note that the optimum for the generator occurs at $D(\mathbf{x}) = D^\star(\mathbf{x}) = 1/2$, so that $Z = 1$ and $\tilde{p}(\mathbf{x}) = p_g(\mathbf{x}) = p(\mathbf{x})$. $\tilde{p}(\mathbf{x})$ is a potentially biased estimator for the true density. However, the bias is implicit only in the quality of $D(\mathbf{x})$: the closer $D(\mathbf{x})$ is to $D^\star(\mathbf{x})$, the lower the bias.

2.3 Discrete variables

The above target distribution reveals a straightforward learning algorithm when $p_g(\mathbf{x})$ is known. For discrete variables, parameterizing the generating distribution directly reveals a means of training the generator without relying on back-propagation through the generator output. As with previous work on evaluating GANs [15], we parameterize $p_g(\mathbf{x})$ as a the marginalization of a joint density, $p_g(\mathbf{x}) = \sum_{\mathbf{z}} g(\mathbf{x}|\mathbf{z})p(\mathbf{z})$, rephrasing the generator function, $G(\mathbf{z})$, as a conditional distribution, $g(\mathbf{x}|\mathbf{z})$. To minimize the distance between $\tilde{p}(\mathbf{x})$ and $p_g(\mathbf{x})$, we can minimize the KL divergence, so that the gradients w.r.t. the generator parameters, $\boldsymbol{\psi}$, are:

$$\begin{aligned}\nabla_{\boldsymbol{\psi}} D_{KL}(\tilde{p}(\mathbf{x}) || p_g(\mathbf{x})) &= \nabla_{\boldsymbol{\psi}} \sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) \log \frac{\tilde{p}(\mathbf{x})}{p_g(\mathbf{x})} \approx - \sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) \nabla_{\boldsymbol{\psi}} \log p_g(\mathbf{x}) \\ &= - \sum_{\mathbf{x}} \frac{1}{Z} p_g(\mathbf{x}) \frac{D(\mathbf{x})}{1 - D(\mathbf{x})} \nabla_{\boldsymbol{\psi}} \log p_g(\mathbf{x}) = - \sum_{\mathbf{x}} \sum_{\mathbf{z}} \frac{1}{Z} g(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) \frac{D(\mathbf{x})}{1 - D(\mathbf{x})} \nabla_{\boldsymbol{\psi}} \log p_g(\mathbf{x}),\end{aligned} \tag{7}$$

where $\tilde{p}(\mathbf{x})$ is held fixed. The gradients would require some sort of Monte-Carlo (MC) estimator across the input and output noise and will have high variance, notably due to the partition function, Z. The intuition here is to note that, as the conditional density, $g(\mathbf{x}|\mathbf{z})$ is unimodal, it can be used to define a unimodal distribution analogous to that of Equation 6. Following this intuition, we propose training the conditional, $g(\mathbf{x}|\mathbf{z})$, to fit a *mode* of the data as defined by:

$$\tilde{p}_{\mathbf{z}}(\mathbf{x}) = \frac{1}{Z_{\mathbf{z}}} g(\mathbf{x}|\mathbf{z}) \frac{D(\mathbf{x})}{1 - D(\mathbf{x})}, \tag{8}$$

where $\tilde{p}_{\mathbf{z}}$ is an estimate of the data in the neighborhood of the mode defined by $\mathbf{z}$, and $Z_{\mathbf{z}} = \sum_{\mathbf{x}} g(\mathbf{x}|\mathbf{z}) \frac{D(\mathbf{x})}{1-D(\mathbf{x})}$ is the marginal that ensures $\tilde{p}_{\mathbf{z}}$ is a proper probability distribution. The gradients then become:

$$\nabla_{\boldsymbol{\psi}} D_{KL}(\tilde{p}_{\mathbf{z}}(\mathbf{x}) || g(\mathbf{z}|\mathbf{x})) \approx -\sum_{\mathbf{x}} \tilde{p}_{\mathbf{z}}(\mathbf{x}) \nabla_{\boldsymbol{\psi}} \log g(\mathbf{x}|\mathbf{z}) \approx -\sum_{m} \tilde{w}^{(m)} \nabla_{\boldsymbol{\psi}} \log g(\mathbf{x}^{(m)}|\mathbf{z}),$$

$$\text{where} \quad \tilde{w}^{(m)} = \frac{w^{(m)}}{\sum_{m'} w^{(m')}} \quad \text{and} \quad w^{(m)} = \frac{D(x^{(m)})}{1 - D(x^{(m)})} \tag{9}$$

are the normalized and unnormalized importance weights respectively. The gradients can be computed over a mini-batch of samples from $p(\mathbf{z})$, so that the updates to the generator parameters, $\boldsymbol{\psi}$, become:

$$\Delta \boldsymbol{\psi} \propto \frac{1}{N} \sum_{n} \sum_{m} \tilde{w}^{(m)} \nabla_{\boldsymbol{\psi}} \log g(\mathbf{x}^{(m)}|\mathbf{z}^{(n)}). \tag{10}$$

This reveals a straightforward procedure for training discrete GANs with a parametric conditional generator without relying on back-propagation. (More here about convergence, other properties, etc)

2.4 Continuous variables

For continuous variables, we could introduce a conditional distribution as we did in the discrete case, defining the importance weights, and defining the gradient at the generator output. However, this would abandon some of the strengths central to GANs relative to other generative models [5]. While $p_g(\mathbf{x})$ is unavailable to us, a simple analysis of Equation 6 shows that the distance (by a number of metrics) between $\tilde{p}(\mathbf{x})$ and $p_g(\mathbf{x})$ is minimized when $\frac{1}{Z} \frac{D(\mathbf{x})}{1-D(\mathbf{x})} = 1$. But, as noted, as the ratio, $\frac{D(\mathbf{x})}{1-D(\mathbf{x})}$, convergences to one, so does the partition function. This reveals an alternative learning objective for the generator as defined at the output of the discriminator, $D(\mathbf{x})$. There are a number of loss functions that would be suitable, and we pick a simple squared-error loss:

$$\mathcal{L}_g(\mathbf{x}) = \frac{1}{2} \left(\log D(\mathbf{x}) - \log(1 - D(\mathbf{x})) \right)^2. \tag{11}$$

(Comments about properties, comparison to normal algorithm, etc)

3 Related Work

Our approach introduces a new learning algorithm that allows for training discrete data with generative adversarial networks (GANs), a first in the literature. Discrete data with GANs is an active area of research, particularly with language model data involving recurrent neural network (RNN) generators. (Talk about some language model GANs here). Our discrete solution is highly analogous to reweighted wake-sleep [RWS, 1], a highly successful approach for training Helmholtz machines with discrete variables.

The Gumbel-max trick is another approach to train Helmholz machines with discrete variables [9], which effectively allows for training as variational autoencoders [VAEs, 10]. However, this trick has only been shown to work in very limited settings, and has yet to been shown to work well with GANs with discrete data. We address this approach more thoroughly in the following section.

4 Results

4.1 Discrete variables

4.1.1 Datasets

4.1.2 Setup

4.1.3 Generation results

4.1.4 Modal analysis?

4.1.5 Re-parameteration methods

4.2 Continuous variables

4.2.1 Datasets

4.2.2 Setup

4.2.3 Generation results

4.2.4 Comparison to normal GANs

5 Conclusion

References

[1] Jörg Bornschein and Yoshua Bengio. Reweighted wake-sleep. *arXiv preprint arXiv:1406.2751*, 2014.

[2] Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

[3] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[4] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[5] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[7] Emil Julius Gumbel and Julius Lieblein. Statistical theory of extreme values and some practical applications: a series of lectures. 1954.

[8] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[9] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[10] Diederik Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[11] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1791–1799, 2014.

[12] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[13] Ruslan Salakhutdinov and Geoffrey E Hinton. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pages 448–455, 2009.

[14] Lawrence K Saul, Tommi Jaakkola, and Michael I Jordan. Mean field theory for sigmoid belief networks. *Journal of artificial intelligence research*, 4(1):61–76, 1996.

[15] Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger Grosse. On the quantitative analysis of decoder-based generative models. *arXiv preprint arXiv:1611.04273*, 2016.