A Connection Between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models

Chelsea Finn[*], **Paul Christiano**[*], **Pieter Abbeel**, **Sergey Levine**
University of California, Berkeley
{cbfinn,paulfchristiano,pabbeel,svlevine}@eecs.berkeley.edu

Abstract

Generative adversarial networks (GANs) are a recently proposed class of generative models in which a generator is trained to optimize a cost function that is being simultaneously learned by a discriminator. While the idea of learning cost functions is relatively new to the field of generative modeling, learning costs has long been studied in control and reinforcement learning (RL) domains, typically for imitation learning from demonstrations. In these fields, learning the cost function underlying observed behavior is known as inverse reinforcement learning (IRL) or inverse optimal control. While at first the connection between cost learning in RL and cost learning in generative modeling may appear to be a superficial one, we show in this paper that certain IRL methods are in fact mathematically equivalent to GANs. In particular, we demonstrate an equivalence between a sample-based algorithm for maximum entropy IRL and a GAN in which the generator's density can be evaluated and is provided as an additional input to the discriminator. Interestingly, maximum entropy IRL is a special case of an energy-based model. We discuss the interpretation of GANs as an algorithm for training energy-based models, and relate this interpretation to other recent work that seeks to connect GANs and EBMs. By formally highlighting the connection between GANs, IRL, and EBMs, we hope that researchers in all three communities can better identify and apply transferable ideas from one domain to another, particularly for developing more stable and scalable algorithms: a major challenge in all three domains.

1 Introduction

GAN에서 D가 학습되어가는 과정을 cost 학습으로 해석한다.

Generative adversarial networks (GANs) are a recently proposed class of generative models in which a generator is trained to optimize a cost function that is being simultaneously learned by a discriminator [8]. While the idea of learning objectives is relatively new to the field of generative modeling, learning cost or reward functions has long been studied in control [5] and was popularized in 2000 for reinforcement learning problems [15]. In these fields, learning the cost function underlying demonstrated behavior is referred to as inverse reinforcement learning (IRL) or inverse optimal control (IOC). At first glance, the connection between cost learning in RL (IRL) and cost learning for generative (GAN에서 D) models may appear to be superficial; however, if we apply GANs to a setting where the generator density can be efficiently evaluated, the result is exactly equivalent to a sample-based algorithm for maximum entropy (MaxEnt) IRL. Interestingly, as MaxEnt IRL is an energy-based model, this connection suggests a method for using GANs to train a broader class of energy-based models.

MaxEnt IRL is a widely-used objective for IRL, proposed by Ziebart et al. [27]. Sample-based algorithms for performing maximum entropy (MaxEnt) IRL have scaled cost learning to scenarios

[*] Indicates equal contribution.

with unknown dynamics, using nonlinear function classes, such as neural networks [4, 11, 7]. We show that the gradient updates for the cost and the policy in these methods can be viewed as the updates for the discriminator and generator in GANs, under a specific form of the discriminator. The key difference to a generic discriminator is that we need to be able evaluate the density of the generator, which we integrate into the discriminator in a natural way.

Traditionally, GANs are used to train generative models for which it is not possible to evaluate the density. When it is possible to evaluate the density, for example in an autoregressive model, it is typical to maximize the likelihood of the data directly. By considering the connection to IRL, we find that GAN training may be appropriate even when density values are available. For example, suppose we are interested in modeling a complex multimodal distribution, but our model does not have enough capacity to represent the distribution. Then maximizing likelihood will lead to a distribution which "covers" all of the modes, but puts most of its mass in parts of the space that have negligible density under the data distribution. These might be images that look extremely unrealistic, nonsensical sentences, or suboptimal robot behavior. A generator trained adversarially will instead try to "fill in" as many of modes as it can, without putting much mass in the space between modes. This results in lower diversity, but ensures that samples "look like" they could have been from the original data.

우리가 모델링하려는 함수가 multimodal인데 우리의 모델의 표현력이 부족하면,

실제분포
모델링한 결과

이런 지역에 많은 prob mass를 할당한다

⇓

모델링으로 sample하면 이상한 그림이 나온다.

⇓

Behavior cloning도 마찬가지다!!

By drawing an exact correspondence between adaptive, sample-based algorithms for MaxEnt IRL and GAN training, we show that this phenomenon occurs and is practically important: GAN training can significantly improve the quality of samples even when the generator density can be exactly evaluated. This is precisely analogous to the observed ability of inverse reinforcement learning to imitate behaviors that cannot be successfully learned through behavioral cloning [21], direct maximum likelihood regression to the demonstrated behavior.

IRL이 BC보다 잘된다.

Interestingly, the maximum entropy formulation of IRL is a special case of an energy-based model (EBM) [26]. The learned cost in MaxEnt IRL corresponds to the energy function, and is trained via maximum likelihood. Hence, we can also show how a particular form of GANs can be used to train EBMs. Recent works have recognized a connection between EBMs and GANs [12, 25]. In this work, we particularly focus on EBMs trained with maximum likelihood, and expand upon the connection recognized by Kim & Bengio [12] for the case where the generator's density can be computed. By formally highlighting the connection between GANs, IRL, and EBMs, we hope that researchers in all three areas can better identify and apply transferable ideas from one domain to another.

2 Background

In this section, we formally define generative adversarial networks (GANs), energy-based models (EBMs), and inverse reinforcement learning (IRL), and introduce notation.

2.1 Generative Adversarial Networks

(간!)

Generative adversarial networks are an approach to generative modeling where two models are trained simultaneously: a generator G and a discriminator D. The discriminator is tasked with classifying its inputs as either the output of the generator, or actual samples from the underlying data distribution $p(\mathbf{x})$. The goal of the generator is to produce outputs that are classified by the discriminator as coming from the underlying data distribution [8].

Formally, the generator takes noise as input and outputs a sample $\mathbf{x} \sim G$, while the discriminator takes as input a sample $\mathbf{x}$ and outputs the probability $D(\mathbf{x})$ that the sample was from the data distribution. The discriminator's loss is the average log probability it assigns to the correct classification, evaluated on an equal mixture of real samples and outputs from the generator:

$$\mathcal{L}_{\text{discriminator}}(D) = \mathbb{E}_{\mathbf{x}\sim p}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim G}[-\log(1 - D(\mathbf{x}))].$$

P에서 나온건 1로, G에서 나온건 0이 되게 D를 학습시킨다.

The generator's loss can be defined one of several similar ways. The simplest definition, originally proposed in [8], is simply the opposite of the discriminator's loss. However, this provides very little training signal if the generator's output can be easily distinguished from the real samples. It is common to instead use the log of the discriminator's confusion [8]. We will define the generator's loss as the sum of these two variants:

$$\mathcal{L}_{\text{generator}}(G) = \underbrace{\mathbb{E}_{\mathbf{x}\sim G}[-\log D(\mathbf{x})]}_{\text{원래 간에는 이것만 있었음}} + \mathbb{E}_{\mathbf{x}\sim G}[\log(1 - D(\mathbf{x}))].$$

여튼 둘다 D(x)가 1이 되도록 하는 x가 나오게 G를 학습

2.2 Energy-Based Models

Energy-based models [14] associate an energy value $E_\theta(\mathbf{x})$ with a sample $\mathbf{x}$, modeling the data as a Boltzmann distribution:

$$p_\theta(\mathbf{x}) = \frac{1}{Z}\exp(-E_\theta(\mathbf{x})) \tag{1}$$

The energy function parameters θ are often chosen to maximize the likelihood of the data; the main challenge in this optimization is evaluating the partition function Z, which is an intractable sum or integral for most high-dimensional problems. A common approach to estimating Z requires sampling from the Boltzmann distribution $p_\theta(\mathbf{x})$ within the inner loop of learning.

Sampling from $p_\theta(\mathbf{x})$ can be approximated by using Markov chain Monte Carlo (MCMC) methods; however, these methods face issues when there are several distinct modes of the distribution and, as a result, can take arbitrarily large amounts of time to produce a diverse set of samples. Approximate inference methods can also be used during training, though the energy function may incorrectly assign low energy to some modes if the approximate inference method cannot find them [14].

2.3 Inverse Reinforcement Learning

The goal of inverse reinforcement learning is to infer the cost function underlying demonstrated behavior [15]. It is typically assumed that the demonstrations come from an expert who is behaving near-optimally under some unknown cost. In this section, we discuss MaxEnt IRL and guided cost learning, an algorithm for MaxEnt IRL.

내 연구에선 이런 가정을 하지 않는다.

2.3.1 Maximum entropy inverse reinforcement learning

Maximum entropy inverse reinforcement learning models the demonstrations using a Boltzmann distribution, where the energy is given by the cost function c_θ:

$$p_\theta(\tau) = \frac{1}{Z}\exp(-c_\theta(\tau)),$$

이거 구하기 힘들다. $Z = \int \exp(-C(\tau))\, d\tau$

Here, $\tau = \{\mathbf{x}_1, \mathbf{u}_1, \ldots, \mathbf{x}_T, \mathbf{u}_T\}$ is a trajectory; $c_\theta(\tau) = \sum_t c_\theta(\mathbf{x}_t, \mathbf{u}_t)$ is a learned cost function parametrized by θ; $\mathbf{x}_t$ and $\mathbf{u}_t$ are the state and action at time step t; and the partition function Z is the integral of $\exp(-c_\theta(\tau))$ over all trajectories that are consistent with the environment dynamics.[2]

Under this model, the optimal trajectories have the highest likelihood, and the expert can generate suboptimal trajectories with a probability that decreases exponentially as the trajectories become more costly. As in other energy-based models, the parameters θ are optimized to maximize the likelihood of the demonstrations. Estimating the partition function Z is difficult for large or continuous domains, and presents the main computational challenge. The first applications of this model computed Z exactly with dynamic programming [27]. However, this is only practical in small, discrete domains, and is impossible in domains where the system dynamics $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ are unknown.

2.3.2 Guided cost learning

Guided cost learning introduces an iterative sample-based method for estimating Z in the MaxEnt IRL formulation, and can scale to high-dimensional state and action spaces and nonlinear cost functions [7]. The algorithm estimates Z by training a new sampling distribution $q(\tau)$ and using importance sampling:

$$\begin{aligned}\mathcal{L}_{\text{cost}}(\theta) &= \mathbb{E}_{\tau\sim p}[-\log p_\theta(\tau)] = \mathbb{E}_{\tau\sim p}[c_\theta(\tau)] + \log Z \\ &= \mathbb{E}_{\tau\sim p}[c_\theta(\tau)] + \log\left(\mathbb{E}_{\tau\sim q}\left[\frac{\exp(-c_\theta(\tau))}{q(\tau)}\right]\right).\end{aligned}$$

importance sampling estimate의 variance를 최소화시킨다.

학습(?)할 sampling 분포

Guided cost learning alternates between optimizing c_θ using this estimate, and optimizing $q(\tau)$ to minimize the variance of the importance sampling estimate.

[2] This formula assumes that $\mathbf{x}_{t+1}$ is a deterministic function of the previous history. A more general form of this equation can be derived for stochastic dynamics [26]. However, the analysis largely remains the same: the probability of a trajectory can be written as the product of conditional probabilities, but the conditional probabilities of the states $\mathbf{x}_t$ are not affected by θ and so factor out of all likelihood ratios.

q분포를 통해 나오는 trajectory τ가 현재까지 구해진 cost인 cθ(τ)를 낮추도록 한다.

The optimal importance sampling distribution for estimating the partition function $\int \exp(-c_\theta(\tau))d\tau$ is $q(\tau) \propto |\exp(-c_\theta(\tau))| = \exp(-c_\theta(\tau))$. During guided cost learning, the sampling policy $q(\tau)$ is updated to match this distribution by minimizing the KL divergence between $q(\tau)$ and $\frac{1}{Z}\exp(-c_\theta(\tau))$, or equivalently minimizing the learned cost and maximizing entropy.

$$\mathcal{L}_{\text{sampler}}(q) = \mathbb{E}_{\tau\sim q}[c_\theta(\tau)] + \underbrace{\mathbb{E}_{\tau\sim q}[\log q(\tau)]}_{\text{regularizer}} \tag{2}$$

Conveniently, this optimal sampling distribution is the demonstration distribution for the true cost function. Thus, this training procedure results in both a learned cost function, characterizing the demonstration distribution, and a learned policy $q(\tau)$, capable of generating samples from the demonstration distribution.

This importance sampling estimate can have very high variance if the sampling distribution q fails to cover some trajectories τ with high values of $\exp(-c_\theta(\tau))$. Since the demonstrations will have low cost (as a result of the IRL objective), we can address this coverage problem by mixing the demonstration data samples with the generated samples. Let $\mu = \frac{1}{2}p + \frac{1}{2}q$ be the mixture distribution over trajectory roll-outs. Let $p(\tau)$ be a rough estimate for the density of the demonstrations; for example we could use the current model p_θ, or we could use a simpler density model trained using another method. Guided cost learning uses μ for importance sampling[3], with $\frac{1}{2}\tilde{p}(\tau) + \frac{1}{2}q(\tau)$ as the importance weights:

$$\mathcal{L}_{\text{cost}}(\theta) = \mathbb{E}_{\tau\sim p}[c_\theta(\tau)] + \log\left(\mathbb{E}_{\tau\sim\mu}\left[\frac{\exp(-c_\theta(\tau))}{\frac{1}{2}\tilde{p}(\tau) + \frac{1}{2}q(\tau)}\right]\right),$$

2.4 Direct Maximum Likelihood and Behavioral Cloning

A simple approach to imitation learning and generative modeling is to train a generator or policy to output a distribution over the data, without learning a discriminator or energy function. For tractability, the data distribution is typically factorized using a directed graphical model or Bayesian network. In the field of generative modeling, this approach has most commonly been applied to speech and language generation tasks [23, 18], but has also been applied to image generation [22]. Like most EBMs, these models are trained by maximizing the likelihood of the observed data points.

When a generative model does not have the capacity to represent the entire data distribution, maximizing likelihood directly will lead to a moment-matching distribution that tries to “cover” all of the modes, leading to a solution that puts much of its mass in parts of the space that have negligible probability under the true distribution. In many scenarios, it is preferable to instead produce only realistic, highly probable samples, by “filling in” as many modes as possible, at the trade-off of lower diversity. Since EBMs are also trained with maximum likelihood, the energy function in an EBM will exhibit the same moment-matching behavior when it has limited capacity. However, designing a flexible energy function to represent a distribution’s density function is generally much easier than designing a tractable generator with the same flexibility, that can to generate samples without a complex iterative inference procedure. Moreover, once we have a trained energy function, the generator is trained to be mode-seeking, by minimizing the KL divergence between the generator’s distribution and the distribution induced by the energy function. As a result, even if the generator has the same capacity as a generative model trained with direct maximum likelihood, the generator trained with an EBM will exhibit mode-seeking behavior as long as the energy function is more flexible than the generator. Of course, this phenomenon is often achieved at the cost of tractability, as generating samples from an energy function requires training a generator which, in the case of IRL, is forward policy optimization.

In sequential decision-making domains, using direct maximum likelihood is known as behavioral cloning, where the policy is trained with supervised learning to match the actions of the demonstrating agent, conditioned on the corresponding observations. While this approach is simple and often effective for small problems, the moment-matching behavior of direct maximum likelihood can produce particularly ineffective trajectories because of compounding errors. When the policy makes a small mistake, it deviates from the state distribution seen during training, making it more likely to make a mistake again. This issue compounds and eventually, the agent reaches a state far from the

[3] In RL settings, where generating samples requires executing a policy in the real world, such as in robotics, old samples from old generators are typically retained for efficiency. In this case, the density q can be computed using a fusion distribution over the past generator densities.

training distribution and makes a catastrophic error [21]. Generative modeling also faces this issue when generating variables sequentially. A popular approach for handling this involves incrementally sampling more from the model and drawing less from the data distribution during training [21]. This requires that the true data distribution can be sampled from during training, corresponding to a human or algorithmic expert. Bengio et al. proposed an approximate solution, termed scheduled sampling, that does not require querying the data distribution [3]. However, while these approaches alleviate the issue, they do not solve it completely.

3 GANs and IRL

We now show how generative adversarial modeling has implicitly been applied to the setting of inverse reinforcement learning, where the data-to-be-modeled is a set of expert demonstrations. The derivation requires a particular form of discriminator, which we discuss first in Section 3.1. After making this modification to the discriminator, we obtain an algorithm for IRL, as we show in Section 3.2, where the discriminator involves the learned cost and the generator represents the policy.

3.1 A special form of discriminator

GAN 에서 G가 고정된 상태에서 ($G=q(\tau)$) optimal D의 형태

For a fixed generator with a [typically unknown] density $q(\tau)$, the optimal discriminator is the following [8]:

$$D^*(\tau) = \frac{\boxed{p(\tau)}}{p(\tau) + \boxed{q(\tau)}}, \qquad (3)$$

← 실제 분포.

↑ 데이터 생성 분포.

where $p(\tau)$ is the actual distribution of the data.

In the traditional GAN algorithm, the discriminator is trained to directly output this value. When the generator density $q(\tau)$ can be evaluated, the traditional GAN discriminator can be modified to incorporate this density information. Instead of having the discriminator estimate the value of Equation 3 directly, it can be used to estimate $p(\tau)$, filling in the value of $q(\tau)$ with its known value. In this case, the new form of the discriminator D_θ with parameters θ is

$$D_\theta(\tau) = \frac{\boxed{\tilde{p}_\theta(\tau)}}{\tilde{p}_\theta(\tau) + \boxed{q(\tau)}}.$$

← D

← 내가 알고있는 G

GAN에서 IRL로 가는 연결고리.

In order to make the connection to MaxEnt IRL, we also replace the estimated data density with the Boltzmann distribution. As in MaxEnt IRL, we write the energy function as c_θ to designate the learned cost. Now the discriminator's output is:

$$D_\theta(\tau) = \frac{\boxed{\frac{1}{Z}\exp(-c_\theta(\tau))}}{\frac{1}{Z}\exp(-c_\theta(\tau)) + \boxed{q(\tau)}}.$$

← MaxEnt Likelihood

Data Generation

잘 이해가 안된다.

The resulting architecture for the discriminator is very similar to a typical model for binary classification, with a sigmoid as the final layer and $\log Z$ as the bias of the sigmoid. We have adjusted the architecture only by subtracting $\log q(\tau)$ from the input to the sigmoid. This modest change allows the optimal discriminator to be completely independent of the generator: the discriminator is optimal when $\frac{1}{Z}\exp(-c_\theta(\tau)) = p(\tau)$. Independence between the generator and the optimal discriminator may significantly improve the stability of training.

This change is very simple to implement and is applicable in any setting where the density $q(\tau)$ can be cheaply evaluated. Of course this is precisely the case where we could directly maximize likelihood, and we might wonder whether it is worth the additional complexity of GAN training. But the experience of researchers in IRL has shown that maximizing log likelihood directly is not always the most effective way to learn complex behaviors, even when it is possible to implement. As we will show, there is a precise equivalence between MaxEnt IRL and this type of GAN, suggesting that the same phenomenon may occur in other domains: GAN training may provide advantages even when it would be possible to maximize likelihood directly.

3.2 Equivalence between generative adversarial networks and guided cost learning

In this section, we show that GANs, when applied to IRL problems, optimize the same objective as MaxEnt IRL, and in fact the variant of GANs described in the previous section is precisely equivalent to guided cost learning.

Recall that the discriminator's loss is equal to

$$\begin{aligned}\mathcal{L}_{\text{discriminator}}(D_\theta) &= \mathbb{E}_{\tau\sim p}[-\log D_\theta(\tau)] + \mathbb{E}_{\tau\sim q}[-\log(1-D_\theta(\tau))] \\ &= \mathbb{E}_{\tau\sim p}\left[-\log \frac{\frac{1}{Z}\exp(-c_\theta(\tau))}{\frac{1}{Z}\exp(-c_\theta(\tau))+q(\tau)}\right] + \mathbb{E}_{\tau\sim q}\left[-\log \frac{q(\tau)}{\frac{1}{Z}\exp(-c_\theta(\tau))+q(\tau)}\right]\end{aligned}$$

In maximum entropy IRL, the log-likelihood objective is:

$$\mathcal{L}_{\text{cost}}(\theta) = \mathbb{E}_{\tau\sim p}[c_\theta(\tau)] + \log\left(\mathbb{E}_{\tau\sim\frac{1}{2}p+\frac{1}{2}q}\left[\frac{\exp(-c_\theta(\tau))}{\frac{1}{2}\widetilde{p}(\tau)+\frac{1}{2}q(\tau)}\right]\right) \tag{4}$$

$$= \mathbb{E}_{\tau\sim p}[c_\theta(\tau)] + \log\left(\mathbb{E}_{\tau\sim\mu}\left[\frac{\exp(-c_\theta(\tau))}{\frac{1}{2Z}\exp(-c_\theta(\tau))+\frac{1}{2}q(\tau)}\right]\right), \tag{5}$$

where we have substituted $\widetilde{p}(\tau) = p_\theta(\tau) = \frac{1}{Z}\exp(-c_\theta(\tau))$, i.e. we are using the current model to estimate the importance weights.

We will establish the following facts, which together imply that GANs optimize precisely the MaxEnt IRL problem:

1. The value of Z which minimizes the discriminator's loss is an importance-sampling estimator for the partition function, as described in Section 2.3.2.
2. For this value of Z, the derivative of the discriminator's loss with respect to θ is equal to the derivative of the MaxEnt IRL objective.
3. The generator's loss is exactly equal to the cost c_θ minus the entropy of $q(\tau)$, i.e. the MaxEnt policy loss defined in Equation 2 in Section 2.3.2.

Recall that μ is the mixture distribution between p and q. Write $\widetilde{\mu}(\tau) = \frac{1}{2Z}\exp(-c_\theta(\tau)) + \frac{1}{2}q(\tau)$. Note that when θ and Z are optimized, $\frac{1}{Z}\exp(-c_\theta(\tau))$ is an estimate for the density of $p(\tau)$, and hence $\widetilde{\mu}(\tau)$ is an estimate for the density of μ.

3.2.1 Z estimates the partition function

We can compute the discriminator's loss:

$$\mathcal{L}_{\text{discriminator}}(D_\theta) = \mathbb{E}_{\tau\sim p}\left[-\log\frac{\frac{1}{Z}\exp(-c_\theta(\tau))}{\widetilde{\mu}(\tau)}\right] + \mathbb{E}_{\tau\sim q}\left[-\log\frac{q(\tau)}{\widetilde{\mu}(\tau)}\right] \tag{6}$$

$$= \log Z + \mathbb{E}_{\tau\sim p}[c_\theta(\tau)] + \mathbb{E}_{\tau\sim p}[\log\widetilde{\mu}(\tau)] - \mathbb{E}_{\tau\sim q}[\log q(\tau)] + \mathbb{E}_{\tau\sim q}[\log\widetilde{\mu}(\tau)] \tag{7}$$

$$= \log Z + \mathbb{E}_{\tau\sim p}[c_\theta(\tau)] - \mathbb{E}_{\tau\sim q}[\log q(\tau)] + 2\mathbb{E}_{\tau\sim\mu}[\log\widetilde{\mu}(\tau)]. \tag{8}$$

Only the first and last terms depend on Z. At the minimizing value of Z, the derivative of these term with respect to Z will be zero:

$$\begin{aligned}\partial_Z\mathcal{L}_{\text{discriminator}}(D_\theta) &= 0 \\ \frac{1}{Z} &= \mathbb{E}_{\tau\sim\mu}\left[\frac{\frac{1}{Z^2}\exp(-c_\theta(\tau))}{\widetilde{\mu}(\tau)}\right] \\ Z &= \mathbb{E}_{\tau\sim\mu}\left[\frac{\exp(-c_\theta(\tau))}{\widetilde{\mu}(\tau)}\right].\end{aligned}$$

Thus the minimizing Z is precisely the importance sampling estimate of the partition function in Equation 4.

3.2.2 c_θ optimizes the IRL objective

We return to the discriminator's loss as computed in Equation 8, and consider the derivative with respect to the parameters θ. We will show that this is exactly the same as the derivative of the IRL objective.

Only the second and fourth terms in the sum depend on θ. When we differentiate those terms we obtain:

$$\partial_\theta \mathcal{L}_{\text{discriminator}}(D_\theta) = \mathbb{E}_{\tau\sim p}[\partial_\theta c_\theta(\tau)] - \mathbb{E}_{\tau\sim\mu}\left[\frac{\frac{1}{Z}\exp(-c_\theta(\tau))\partial_\theta c_\theta(\tau)}{\widetilde{\mu}(\tau)}\right]$$

On the other hand, when we differentiate the MaxEnt IRL objective, we obtain:

$$\begin{aligned}
\partial_\theta \mathcal{L}_{\text{cost}}(\theta) &= \mathbb{E}_{\tau\sim p}[\partial_\theta c_\theta(\tau)] + \partial_\theta \log\left(\mathbb{E}_{\tau\sim\mu}\left[\frac{\exp(-c_\theta(\tau))}{\widetilde{\mu}(\tau)}\right]\right) \\
&= \mathbb{E}_{\tau\sim p}[\partial_\theta c_\theta(\tau)] + \left(\mathbb{E}_{\tau\sim\mu}\left[\frac{-\exp(-c_\theta(\tau))\partial_\theta c_\theta(\tau)}{\widetilde{\mu}(\tau)}\right] \Big/ \mathbb{E}_{\tau\sim\mu}\left[\frac{\exp(-c_\theta(\tau))}{\widetilde{\mu}(\tau)}\right]\right) \\
&= \mathbb{E}_{\tau\sim p}[\partial_\theta c_\theta(\tau)] - \mathbb{E}_{\tau\sim\mu}\left[\frac{\frac{1}{Z}\exp(-c_\theta(\tau))\partial_\theta c_\theta(\tau)}{\widetilde{\mu}(\tau)}\right] \\
&= \partial_\theta \mathcal{L}_{\text{discriminator}}(D_\theta).
\end{aligned}$$

In the third equality, we used the definition of Z as an importance sampling estimate. Note that in the second equality, we have treated $\widetilde{\mu}(\tau)$ as a constant rather than as a quantity that depends on θ. This is because the IRL optimization is minimizing $\log Z = \log \sum_\tau \exp(-c_\theta(\tau))$ and using $\widetilde{\mu}(\tau)$ as the weights for an importance sampling estimator of Z. For this purpose we do not want to differentiate through the importance weights.

3.3 The generator optimizes the MaxEnt IRL objective

Finally, we compute the generator's loss:

$$\begin{aligned}
\mathcal{L}_{\text{generator}}(q) &= \mathbb{E}_{\tau\sim q}[\log(1-D(\tau)) - \log(D(\tau))] \\
&= \mathbb{E}_{\tau\sim q}\left[\log\frac{q(\tau)}{\widetilde{\mu}(\tau)} - \log\frac{\frac{1}{Z}\exp(-c_\theta(\tau))}{\widetilde{\mu}(\tau)}\right] \\
&= \mathbb{E}_{\tau\sim q}[\log q(\tau) + \log Z + c_\theta(\tau)] \\
&= \log Z + \mathbb{E}_{\tau\sim q}[c_\theta(\tau)] + \mathbb{E}_{\tau\sim q}[\log q(\tau)] = \log Z + \mathcal{L}_{\text{sampler}}(q).
\end{aligned}$$

The term $\log Z$ is a parameter of the discriminator that is held fixed while optimizing the generator, this loss is exactly equivalent the sampler loss from MaxEnt IRL, defined in Equation 2.

3.4 Discussion

There are many apparent differences between MaxEnt IRL and the GAN optimization problem. But, we have shown that after making a single key change—using a generator $q(\tau)$ for which densities can be evaluated efficiently, and incorporating this information into the discriminator in a natural way—generative adversarial networks can be viewed as a sample-based algorithm for the MaxEnt IRL problem. By connecting GANs to the empirical literature on inverse reinforcement learning [7], this demonstrates that GAN training can improve the quality of samples even when the generator's density can be evaluated exactly. By generalizing this connection, we can derive a new adversarial training strategy for energy-based models, which we discuss in the next section.

4 GANs for training EBMs

Now that we have highlighted the connection between GANs and guided cost learning, the application of GANs to EBMs follows directly. As discussed in Section 2.2, the primary challenge in training EBMs is estimating the partition function, which is done by approximately sampling from the distribution induced by the energy E_θ. Two recent papers have proposed to use adversarial training to derive fast estimates of the partition function [12, 25]. In particular, these methods alternate between training a generator to produce samples with minimal energy $E_\theta(\mathbf{x})$, and optimizing the parameters of the energy function using the samples to estimate the partition function.

When the density of the generator is available, however, we can derive an unbiased estimate of the partition function as

$$Z = \mathbb{E}_{\mathbf{x}\sim\mu}\left[\frac{\exp(-E_\theta(\mathbf{x}))}{\frac{1}{2}\widetilde{p}(\mathbf{x}) + \frac{1}{2}q(\mathbf{x})}\right]$$

where μ denotes an equal mixture of generated and real data points, $q(\mathbf{x})$ denotes the density under the generator, and $\widetilde{p}(\mathbf{x})$ denotes an estimate for the data density.

This gives a loss function

$$\begin{aligned}\mathcal{L}_{\text{energy}}(\theta) &= \mathbb{E}_{\mathbf{x}\sim p}[-\log p_\theta(\mathbf{x})]\\ &= \mathbb{E}_{\mathbf{x}\sim p}[-E_\theta(\mathbf{x})] - \log\left(\mathbb{E}_{\mathbf{x}\sim\mu}\left[\frac{\exp(-E_\theta(\mathbf{x}))}{\frac{1}{2}\widetilde{p}(\mathbf{x}) + \frac{1}{2}q(\mathbf{x})}\right]\right).\end{aligned}$$

As before, the generator is updated to minimize energy and maximize entropy:

$$\mathcal{L}_{\text{generator}}(q) = \mathbb{E}_{\mathbf{x}\sim q}[E_\theta(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim q}[\log q(\mathbf{x})]$$

If we set $\widetilde{p}(\mathbf{x}) = p_\theta(\mathbf{x})$, the resulting model is a special case of a GAN which is straightforward to implement. The discriminator's output is $\sigma(E_\theta(\mathbf{x}) - \log q(\mathbf{x}))$, where σ is a sigmoid with a trainable bias. The discriminator's loss is the log probability and the generator's loss is the discriminator's log odds, as defined in Section 2.1.

Kim & Bengio proposed a similar energy-based model for generative image modeling, but did not assume they could compute the generator's density [12]. As a result, they do not use importance weights, and work with a biased estimator of the partition function which converges to the true partition function when the generator correctly samples from the energy-based model. In contrast, by using the generator density, we can get an unbiased estimate of the partition function that does not rely on any assumptions about the generator. Thus, even if the generator cannot learn to sample exactly from the data distribution, our training procedure is consistent.

Zhao et al. also proposed an energy-based GAN model with an autoencoder discriminator where the energy is given by the mean-squared error between the data example (generated or real) and the discriminator's reconstruction [25]. The energy function is optimized with a margin loss, and the generator is trained to minimize energy. This method also did not use the form of discriminator presented above. An interesting direction for future exploration is to consider combining the GAN training algorithm discussed here with an objective other than log-likelihood, such as one used with EBMs [14] or different f-divergences used with GANs [17].

5 Related Work

Ho et al. [10, 9] previously presented a GAN-like algorithm for imitation learning, where the goal is to recover a policy that matches the expert demonstrations. The proposed algorithm, called generative adversarial imitation learning (GAIL), has an adversarial structure. The analysis in this paper provides additional insight into what GAIL is doing. As discussed above, GANs are optimizing the same objective as MaxEnt IRL. Thus, the GAIL policy is being trained to optimize a cost learned through MaxEnt IRL. Unlike guided cost learning [7], however, Ho & Ermon use the typical unconstrained form of the discriminator [9] and do not use the generator's density. In this case, the cost function remains implicit within the discriminator and cannot be recovered. Hence, in GAIL, the discriminator is discarded and the policy is the end result.

Bachman & Precup [1] suggested that data generation can be converted into a sequential decision-making problem and solved with a reinforcement learning method. Several recent works have proposed methods for merging maximum likelihood objectives and known reward functions for training sequential language generation models and rely on surrogate reward function such as BLEU score or edit distance [20, 16, 2]. In this work, we assume that the reward function is unknown.

Yu et al. proposed to learn a cost function for sequential data generation using GANs, where the cost is defined as the probability of the discriminator classifying the generated sequence as coming from the data distribution [24]. The discriminator does not take advantage of the policy's density values, despite the fact that they are known (and are used during pre-training). Their experiments also find that max-likelihood pre-training is crucial for good performance, suggesting that recurrent generators that can't afford such pre-training (e.g. because they don't have densities) are less practical to train.

Pfau & Vinyals drew a connection between the optimization problems in GANs and actor-critic methods in reinforcement learning, suggesting how ideas for stabilizing training in one domain could be beneficial for the other [19]. As the authors point out, these optimization tricks could also be useful for imitation learning algorithms with the same two-level optimization structure.

6 Discussion

In this work, we showed an equivalence between generative adversarial modeling and an algorithm for performing maximum entropy inverse reinforcement learning. Our derivation used a special form of discriminator that leverages likelihood values from the generator, leading to an unbiased estimate of the underlying energy function. A natural direction for future work is to experiment with combining deep generators that can provide densities, such as autoregressive models [13, 22] or models that use invertible transformations [6], with generative adversarial modeling. Such an approach may provide more stable training, better generators, and wider applicability to discrete problems such as language.

This work also suggests a new algorithm for training energy-based models using generative adversarial networks, that trains a neural network model to sample from the distribution induced by the current energy. This method could reduce the computational challenges of existing MCMC-based solutions.

Acknowledgments

The authors would like to thank Ian Goodfellow and Joan Bruna for insightful discussions.

References

[1] P. Bachman and D. Precup. Data generation as sequential decision making. In *Neural Information Processing Systems (NIPS)*, 2015.

[2] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, 2016.

[3] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Neural Information Processing Systems (NIPS)*, 2015.

[4] A. Boularias, J. Kober, and J. Peters. Relative entropy inverse reinforcement learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[5] S. Boyd and L. Vandenberghe. Convex optimization, 2004.

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

[7] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. *International Conference on Machine Learning (ICML)*, 2016.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Neural Information Processing Systems (NIPS)*, 2014.

[9] J. Ho and S. Ermon. Generative adversarial imitation learning. *Neural Information Processing Systems (NIPS)*, 2016.

[10] J. Ho, J. K. Gupta, and S. Ermon. Model-free imitation learning with policy optimization. In *International Conference on Machine Learning (ICML)*, 2016.

[11] M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal. Learning objective functions for manipulation. In *International Conference on Robotics and Automation (ICRA)*, 2013.

[12] T. Kim and Y. Bengio. Deep directed generative models with energy-based probability estimation. *ICLR Workshop Track*, 2016.

[13] H. Larochelle and I. Murray. The neural autoregressive distribution estimator. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[14] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. A tutorial on energy-based learning. *Predicting structured data*, 1:0, 2006.

[15] A. Ng, S. Russell, et al. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2000.

[16] M. Norouzi, S. Bengio, Z. Chen, N. Jaitly, M. Schuster, Y. Wu, and D. Schuurmans. Reward augmented maximum likelihood for neural structured prediction. *Neural Information Processing Systems (NIPS)*, 2016.

[17] S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Neural Information Processing Systems (NIPS)*, 2016.

[18] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[19] D. Pfau and O. Vinyals. Connecting generative adversarial networks and actor-critic methods. *arXiv preprint arXiv:1610.01945*, 2016.

[20] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. *International Conference on Learning Representations (ICLR)*, 2016.

[21] S. Ross, G. Gordon, and A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *Journal of Machine Learning Research*, 15, 2011.

[22] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *International Conference on Machine Learning (ICML)*, 2016.

[23] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[24] L. Yu, W. Zhang, J. Wang, and Y. Yu. Seqgan: Sequence generative adversarial nets with policy gradient. *arXiv preprint arXiv:1609.05473*, 2016.

[25] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

[26] B. Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, Carnegie Mellon University, 2010.

[27] B. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2008.