Notes On GANs, Energy-Based Models, and Saddle Points

John Schulman joschu@openai.com

# 1 Introduction

These notes explore a family of methods related to generative adversarial networks (GANs) [Goo+14]; these methods try to estimate a probability distribution from samples, using a minimax objective that involves a generator/sampler and discriminator/cost. First we'll derive a minimax expression for negative log-likelihood of energy-based models. Reviewing an idea from [HE16], we'll show that after adding a particular regularizer to the cost function, we can recover the original GAN objective, drawing a close connection between energy-based models and GANs. The objectives we consider are convex-concave functions of the cost and sampling density (but not necessarily with respect to their parameters). Convex-concave problems are tractable (like convex optimization problems) and we discuss some results from the theory that may provide useful intuitions.

## Q & A

- *The connections between energy-based models and GANs have already been pointed out in [KB16] and [Chr16]. How is this writeup different?* Those papers show that the gradient expressions for GANs are **almost** the same as the gradient expressions for energy-based models. That's a useful observation, but it provides an incomplete picture of what the algorithms converge to after many updates. This writeup focuses on how the **objective functions** are similar and different.

- *We already know from [Goo+14] that GANs converge "in function space"—i.e., when we assume that the discriminator is optimized over the space of all functions each iteration. Given that your analysis relies on the convex-concave property in function space, what does it add?* The limitation of the convergence anlaysis in [Goo+14] is that the actual optimization procedure for GANs **doesn't** fully optimize the discriminator each iteration, it simultaneously performs small updates to both the generator and discriminator. This procedure does not converge in general for minimax problems $\min_x \max_y f(x, y)$. However, it turns out that for convex-concave problems (i.e., where $f$ is convex in $x$ and concave in $y$), gradient-based updates do converge to an optimal point (called an equilibrium or saddle point) where $\min_x \max_y f(x, y) = \max_y \min_x f(x, y)$.

# 2 Energy-Based Models

Energy based models define a probability distribution in terms of a cost (energy) function $c$:

$$p(x) = e^{-c(x)}/Z_c, \tag{1}$$

$$\text{where } Z_c = \int \mathrm{d}x \; e^{-c(x)} \tag{2}$$

Given a set of datapoints $x_1, x_2, \ldots, x_N$, the total negative log-likelihood is

$$\text{total nll} = \sum_{n=1}^{N} -\log p(x_n) = \sum_{n=1}^{N} (c(x_n) + \log(Z_c)) \tag{3}$$

It is more meaningful to consider the loss per datapoint, which is

$$\text{nll per datapoint} = \frac{1}{N}(\text{total nll}) = \mathbb{E}_{\text{data}}\left[c(x)\right] + \log(Z_c). \tag{4}$$

where $\mathbb{E}_{\text{data}}\left[\ldots\right]$ indicates that we sample $x$ uniformly from $x_1, x_2, \ldots, x_N$.

We can write down an importance sampling estimator for $Z$, as an expectation under a distribution $q(x)$.

$$Z_c = \int \mathrm{d}x \; e^{-c(x)} = \int \mathrm{d}x \; q(x) \frac{e^{-c(x)}}{q(x)} = \mathbb{E}_q \left[ \frac{e^{-c(x)}}{q(x)} \right] \tag{5}$$

Thus,

$$\text{nll per datapoint} = \mathbb{E}_{\text{data}}\left[c(x)\right] + \log \mathbb{E}_q \left[ \frac{e^{-c(x)}}{q(x)} \right] \tag{6}$$

Jensen's inequality implies that $\log(\mathbb{E}\left[y\right]) \geq \mathbb{E}\left[\log(y)\right]$, with equality where $y$ is constant. Thus,

$$\begin{aligned} \text{nll per datapoint} &\geq \mathbb{E}_{\text{data}}\left[c(x)\right] + \mathbb{E}_q \left[ \log \frac{e^{-c(x)}}{q(x)} \right] && (7) \\ &= \mathbb{E}_{\text{data}}\left[c(x)\right] - \mathbb{E}_q\left[c(x)\right] - \mathbb{E}_q\left[\log q(x)\right] && (8) \\ &= \mathbb{E}_{\text{data}}\left[c(x)\right] - \mathbb{E}_q\left[c(x)\right] + H(q) && (9) \end{aligned}$$

Hence, the right-hand side expression is a lower bound on bound on the negative log-likelihood. Equality holds when $q(x) = e^{-c(x)}/Z_c$, hence we can write

$$\text{nll per datapoint} = \max_q \left( \mathbb{E}_{\text{data}}\left[c(x)\right] - \mathbb{E}_q\left[c(x)\right] + H(q) \right) \tag{10}$$

We are minimizing negative log-likelihood, so our full optimization problem looks like

$$\min_c \left( \text{nll per datapoint} \right) = \min_c \max_q \left( \mathbb{E}_{\text{data}}\left[c(x)\right] - \mathbb{E}_q\left[c(x)\right] + H(q) \right) \tag{11}$$

In summary, we wrote the likelihood maximization problem (wrt $c$) as a minimax problem, involving a sampling distribution $q$.r This could be turned into an algorithm, which jointly learns a sampling distribution $q$ and the cost (energy) function $c$. In practice, when $c$ and $q$ are represented by nonlinear function approximators, we will need to jointly optimize them by SGD, thus $q$ will not have a chance to fully catch up with $c$. With this formulation $c$ can grow without bound, so the minimax objective above may behave unstably.

# 3 Cost Regularization

Let's introduce a regularization term $\psi(c)$ which encourages $c$ to be small. This change can be interpreted as introducing a prior on $c$. Since $c$ encodes a probability distribution, we'd prefer for it to be smooth and simple, rather than putting a delta function at the data points. The objective is redefined as follows, to be the nll plus a regularization term:

$$L(c) = \mathbb{E}_{\text{data}}\left[c(x)\right] + \log(Z_c) + \psi(c) \tag{12}$$

Repeating the derivation of the previous section (but with $\psi(c)$ added), we get

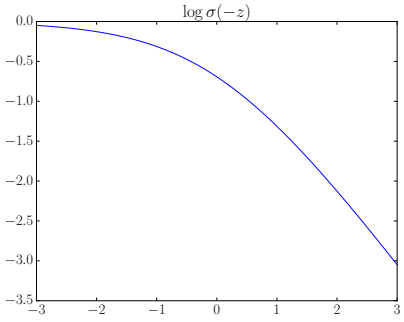
$$L(c) = \max_q \left( \mathbb{E}_{\text{data}}\left[c(x)\right] - \mathbb{E}_q\left[c(x)\right] + H(q) + \psi(c) \right) \tag{13}$$

$$= \max_q L(c, q) \tag{14}$$

where the last line is the definition of $L(c, q)$.

## 3.1 Deriving the GAN Objective

Now we'll show that by reparameterizing $c$ and choosing a particular regularizer $\psi(c)$, we can derive the original GAN objective, plus the entropy term $H(q)$. Let $c(x) = \log\sigma(-f(x))$, where $\sigma$ is the sigmoid function $\sigma(z) = 1/(1+e^{-z})$, and $f$ is a function approximator with a scalar output, e.g., the output of a neural network, whose last layer has output size 1 and linear activation. (Note that with this definition, large $f \leftrightarrow$ low cost $\leftrightarrow$ the sample looks like it came from $p_{\text{data}}$.) This function is plotted below.



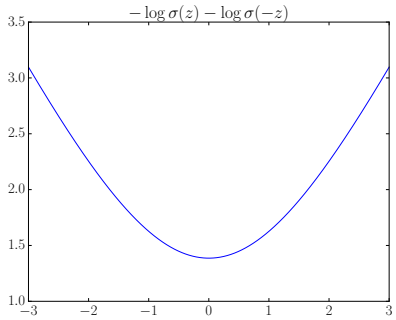

Previously we referred to $\psi(c)$, but now $c$ is defined in terms of $f$, so we'll write $\psi(f)$ for the regularization term. The objective becomes

$$L(f, q) = \mathbb{E}_{\text{data}}\left[\log\sigma(-f(x))\right] - \mathbb{E}_q\left[\log\sigma(-f(x))\right] + H(q) + \psi(f) \tag{15}$$

We're going to define $\psi(f)$ as the following expectation over the data:

$$\psi(f) = \mathbb{E}_{\text{data}}\left[-\log\sigma(f(x)) - \log\sigma(-f(x))\right] \tag{16}$$

We designed this term so it would cancel out the $\log\sigma(-f(x))$ and replace it with a $-\log\sigma(f(x))$. The regularizer $-\log\sigma(z) - \log\sigma(-z)$ is plotted below. It's $\approx z^2 + 2\log 2$ around the origin but then becomes $\approx |z|$ as $|z| \to \infty$.




$$\begin{aligned} L(f,q) &= \mathbb{E}_{\text{data}}\left[\log\sigma(-f(x))\right] - \mathbb{E}_q\left[\log\sigma(-f(x))\right] + H(q) + \mathbb{E}_{\text{data}}\left[-\log\sigma(f(x)) - \log\sigma(-f(x))\right] \\ &= -\mathbb{E}_{\text{data}}\left[\log\sigma(f(x))\right] - \mathbb{E}_q\left[\log\sigma(-f(x))\right] \end{aligned} \tag{17}$$

The sigmoid function has the nice property that

$$\text{sigmoid}(-z) = \frac{1}{1+e^z} = 1 - \frac{e^z}{1+e^z} = 1 - \text{sigmoid}(z). \tag{18}$$

Thus we get

$$L(f,q) = -\mathbb{E}_{\text{data}}\left[\log\sigma(f(x))\right] - \mathbb{E}_q\left[\log(1-\sigma(f(x)))\right] + H(q) \tag{19}$$

$$= -\mathbb{E}_{\text{data}}\left[\log(D(x))\right] - \mathbb{E}_q\left[\log(1-D(x))\right] + H(q) \tag{20}$$

$$\text{defining } D(x) = \sigma(f(x))$$

and our optimization problem is

$$\min_f \max_q L(f,q) \tag{21}$$

There are two difference between the optimization problem we've derived in this section and the original GAN formulation of [Goo+14].

1. The entropy regularization, $H(q)$
2. The min and the max are switched—in the original GAN formulation, the generator is on the outside, but here, the generator is on the inside.

**Why does it make sense to switch the min and max?** When we add the entropy regularization term, we can freely switch the min and the max, both orderings yield the same solution $(f^*, q^*)$. That follows from the properties of convex-concave functions, which are discussed in Section 4, and specialized to the GAN case in Section 5.

**One ugly detail—normalization.** There is one problem with the entropy-regularized formulation. Since cost is parameterized as $c(x) = \log\sigma(-f(x))$, we have that $c(x) \leq 0$. If the domain of $x$ is infinite, then $e^{-c(x)}$ will have an infinite integral, i.e., we won't have a finite partition function. The underlying issue is that on an infinite space, the entropy regularization is too strong—the "pressure" to spread out $q$ is stronger than the pressure to stay near the low-cost regions $c(x)$. This problem can be fixed by using a KL divergence penalty $-K(q, q_0)$ instead of the entropy bonus, where, $q_0$ could be a Gaussian distribution covering the range of reasonable values for $x$.

# 4 Saddle Points

This section provides a general discussion of the optimization problems we've encountered above, which involve a minimization over one set of variables and a maximization over the others. This section will make it possible to answer questions such as "what happens if we switch the min and max?" and "does gradient descent converge to the solution of a minimax problem?" We'll start out by defining three key concepts: minimax problems, saddle points, and convex-concave problems.

1. *Minimax problems* are optimization problems that take the form $\min_x \max_y f(x, y)$. In general, the min and the max do not commute: it's not true in general that the value $\min_x \max_y f(x, y) = \max_y \min_x f(x, y)$, and it's not true in general that a solution $(x^*, y^*)$ to one ordering will be a solution to the other. (When we say $(x^*, y^*)$ is a solution for the ordering $\min_x \max_y f(x, y)$, we mean that $x^*$ minimizes $\max_y f(x, y)$, and $y^* \in \operatorname{argmax}_y f(x^*, y)$.)

2. A *saddle point* is defined as a pair $(x^*, y^*)$ satisfying $x^* \in \operatorname{argmin}_x f(x, y^*)$, and $y^* \in \operatorname{argmax}_y f(x^*, y)$. That is, we can exchange the min and max.

3. A *convex-concave* function $f(x, y)$ is convex in its first argument, and concave in its second argument.

A basic theorem states that if $f$ is convex-concave, then we can always exchange the min and max and the value is unchanged: $\min_x \max_y f(x, y) = \max_y \min_x f(x, y)$. Furthermore, for convex-concave function, any point where the gradient vanishes, $\nabla_x f(x, y) = \nabla_y f(x, y) = 0$, is a saddle point.

Minimax problems have an interpretation as a two player game between Xander and Yasaman. $x$ is Xander's move, and $y$ is Yasaman's move. Xander is trying to minimize $f(x, y)$, and Yasaman is trying to maximize it. The ordering $\min_x \max_y f(x, y)$ means that Xander goes first, whereas $\max_y \min_x f(x, y)$ means that Yasaman goes first. The second player has an advantage, because he or she can see the first player's move and respond accordingly—that is, $\max_y \min_x f(x, y) \leq \min_x \max_y f(x, y)$.

Finding saddle points of convex-concave functions is tractable, unlike solving general minimax problems. In a way, finding these saddle points is on the same level of hardness as finding the minimizers of convex functions. In fact, much of the theory for analyzing (stochastic) gradient descent carries over from convex minimization to the problem of finding saddle points.

Saddle points play a key role in constrained convex optimization problems, where the solution corresponds to finding the saddle point of the Lagrangian $L(x, \lambda, \nu)$, which is convex in the argument $x$ and concave in the Lagrange multipliers $(\lambda, \nu)$. We can find the saddle point using Newton's method. (See [BV04], 10.3.)

One issue that makes saddle point problems harder to understand than minimization problems is that it's less straightforward to measure optimization progress. For minimization problems $\min_x f(x)$, we can trivially measure progress through the objective $f$, which should decrease. For saddle point problems $\min_x \max_y f(x, y)$, we have two ways of measuring progress:

1. **The Gap**. Given a point $(x, y)$, define

$$\operatorname{gap}(x, y) = \max_{y'} f(x, y') - \min_{x'} f(x', y) \tag{22}$$

Recall that the saddle point $(x^*, y^*)$ satisfies

$$f(x^*, y^*) = \min_x \max_y f(x, y) = \max_y \min_x f(x, y) \tag{23}$$

so $\text{gap}(x^*, y^*) = 0$. For arbitrary $(x, y)$, $\text{gap}(x, y) \geq 0$. Most of the convergence theory of gradient descent methods for convex-concave problems relies on showing that the gap is small after optimization.

2. **Gradient Norm**. Another measure of convergence is given by the norm of the gradient with respect to $x$ and $y$: $\|\nabla_x f(x, y)\|^2 + \|\nabla_y f(x, y)\|^2$. The most effective algorithms for solving constrained convex optimization problems are primal-dual methods, which perform Newton steps on the Lagrangian. These methods typically perform line searches on this gradient norm, called the primal-dual residual [BV04].

There are several different techniques used to prove and analyze the convergence of gradient descent in convex-concave problems. Convergence for saddle point problems is less intuitively clear than for convex minimization problems, so these proof techniques might provide some helpful intuitions.

1. *Show that gradient norm is reduced along the step direction.* Let $z = \begin{pmatrix} x \\ y \end{pmatrix}$, and we'll write $f(z)$ to mean $f(x, y)$. Consider taking a small step $z \to z + a$. Taking a first-order Taylor expansion

$$f'(z + a) = f'(z) + f''(z)a + O(\|a\|^2) \tag{24}$$

   (a) Second order methods compute the step that solves $f''(z)a = -f'(z)$, e.g., see the discussion on the infeasible-start Newton method in [BV04]. They perform a line search in this direction, which is **guaranteed** to reduce the gradient norm.

   (b) For large-scale applications, we're more interested in first-order methods, which take a step in the gradient direction. Let $a = -\alpha \begin{pmatrix} \frac{\partial}{\partial x} f(x, y) \\ -\frac{\partial}{\partial y} f(x, y) \end{pmatrix} = -\alpha S f'(z)$ where we define $S = \begin{pmatrix} I & 0 \\ 0 & -I \end{pmatrix}$. Substituting back into Equation (24),

$$f'(z + a) = f'(z) - \alpha f''(z) S f'(z) = (I - \alpha H S) f'(z) \tag{25}$$

   where $H = f''(z)$ is the Hessian of $f$. If we require that $f$ is strongly convex wrt $x$ and strongly concave wrt $y$, the $HS$ is positive definite, and for small $\alpha$, $\|(I - \alpha H S) f'(z)\| < \|f'(z)\|$, so the norm of the gradient strictly decreases.

2. *Online gradient descent / online mirror descent.* The standard analysis from online learning mostly carries through. See [Bub].

3. *Standard subgradient descent convergence analysis.* [NO09] provide a convergence analysis that looks like the standard convergence proofs for subgradient descent, which is also quite similar to the online learning results.

4. *Theory of monotone operators.* One can show that the subgradient update is a contraction using a general and elegant theory of monotone operators [RB16].

Unfortunately it's not the case that the gap monotonically decreases during gradient descent, rather, the gradient norm decreases and the gap reduces as a result.

### Simple Examples

The following two-dimensional problem illustrates some of the properties of the GAN problem.

$$\min_x \max_y x^2 - y(x-1) \tag{26}$$

Here, $y$ corresponds to the generator, and $x$ to the discriminator. The problem is convex-concave and has a saddle point at $(1, 2)$, which one can see by solving for $\nabla_x f(x, y) = \nabla_y f(x, y) = 0$. For each fixed value of $y$, there is a unique solution for $x$. But for each fixed value of $x \neq 1$, the objective is unbounded in $y$, and for $x = 1$, all values $y$ achieve the optimum. The objective above is the Lagrangian of the problem

$$\min_x x^2, \text{ subject to } x = 1 \tag{27}$$

Now consider adding regularization to $y$:

$$\min_x \max_y x^2 - y(x-1) - \epsilon y^2 \tag{28}$$

This problem has a different saddle point: $\left(\frac{1}{1-4\epsilon}, \frac{2}{1-4\epsilon}\right)$. With this objective, if we fix $x$ and optimize over $y$, there is always a unique solution—we can always recover $y$ from $x$. However, if $\epsilon$ is too large, the saddle point goes to infinity.

## 5 GANs and Saddles

Recall the GAN optimization problem, and the entropy-regularized version, which we derived as a likelihood maximization problem.

$$\max_q \min_f \Big( -\mathbb{E}_{\text{data}} \left[\log(\sigma(f(x)))\right] - \mathbb{E}_q \left[\log(1 - \sigma(f(x)))\right] \quad [\,+ H(q)]\Big) \tag{29}$$

The training procedure for GANs is to perform gradient descent on $f$ and $q$ simultaneously. Thus, the training procedure is agnostic to the ordering of the min and max. The key questions are (1) what should this training procedure converge to, and (2) under what conditions does it actually converge? These questions are nontrivial even when optimizing in function space, e.g., with tabular representations of $f$ and $q$.

Let's consider the cases with and without entropy regularization.

### Without Entropy Regularization

- *Saddle.* $(f^*, q^*) = (0, p_{\text{data}})$ is a saddle point, since it satisfies $q^* \in \text{argmax}_q L(q, f^*)$ and $f^* \in \text{argmin}_f L(q^*, f)$.
- *Discriminator on inside.* If the discriminator is the inner optimization problem, i.e., if we solve for $\max_q \min_f L(q, f)$, then for every fixed generator, there's a unique optimal solution for the discriminator $f$. If the discriminator ranges over all functions, then it's the odds ratio $f(x) = p_{\text{data}}(x)/(p_{\text{data}}(x) + q(x))$.

- *Generator on inside.* If we take the generator to be the inner optimization problem, then for a fixed discriminator, the problem $L(f, q)$ may have multiple solutions. If $f$ has a unique global maximum, then $\operatorname{argmax}_q L(f, q)$ is a delta function at the global maximum of $f$. If $f$ is constant (as in the optimal solution), then all functions $q$ are maximizers.

Hence, while the ordering $\min_f \max_q$ does yield the same optimal value as $\max_q \min_f$, this ordering has some unwholesome properties. In particular, it doesn't satisfy the *recoverability* condition—given $f$, we can't recover $q$. (But given $q$, we can recover $f$.)

### With Entropy Regularization

- *Saddle.* The saddle exists[1], but is different from the saddle point of the unregularized problem and can't be computed in closed form.
- *Discriminator on inside.* The optimal discriminator is the same as in the unregularized case.
- *Generator on inside.* $\max_q L(f, q)$ now has a unique nontrivial solution: $q(x) = e^{-c(x)}/Z_c = e^{-\log\sigma(-f(x))}/Z_c = \frac{1}{\sigma(-f(x))Z_c} = \frac{1}{(1-D(x))Z_c}$. Thus the *recoverability* property holds—given $f$, we can recover $q$, and vice versa.

### Does the Saddle Point Theory Have Practical Implications for GAN-like Problems?

- It might be possible to use an approximation of the gap as a convergence diagnostic for GANs: perform a small number of generator-only updates and a small number of discriminator-only updates and measure the gap.
- Since we don't have access to the density of the generator, it's not straightforward to approximate its entropy. However, we may be able to devise more tractable regularizers that make the problem convex-concave, and thus make the generator recoverable in terms of the discriminator/cost.
- Convergence guarantees only hold under restrictive assumptions, for example, that the function is convex-concave. The GAN objective is convex-concave in the functions $c$ and $q$, but not in the parameters. However, it may still be possible to obtain a convergent algorithm when optimizing in terms of parameters. Let's suppose we have an algorithm that is guaranteed to converge to the saddle point of a convex-concave problem, and it works by solving a series a subproblems, as with proximal methods and trust region methods. Then we can try to mimic the behavior of this algorithm by solving these subproblems in terms of parameters. Natural gradient algorithms can be derived this way.

## 6 Generalizations: $\phi$-risks and $f$-divergences

There are a couple of interesting generalizations of GANs that have appeared recently. [NCT16] show how the objective can be altered to optimize various $f$-divergences between the generator's distribution and the data distribution. [HE16] uncover a related generalization—that there is a

[1] Given that $x$ is restricted to a finite space, due to the issue we discussed under "One ugly detail".

connection between classification risks ($\phi$-risks), and $f$-divergences—GANs naturally emerge from using a log-loss, but other divergences arise from other losses. Both [NCT16; HE16] build on [NWJ09] where some key mathematical ideas originated. There is a close correspondence between the regularizer $\psi(c)$ (in Equation (13), for example) and the resulting $f$-divergence / $\phi$-risk being minimized.

$$f\text{-divergences} \quad \Leftrightarrow \quad \phi\text{-risks} \quad \Leftrightarrow \quad \text{cost regularizers } \psi(c)$$

### 6.1 $\phi$-risks

Section 3 showed how the difference-of-costs objective in Equation (13) can be converted into the GAN-like objective in Equation (19), after choosing the appropriate regularizer $\psi(c)$. Moreover, minimization wrt the discriminator results in the Jensen-Shannon divergence between $q$ and the data distribution, $\min_q L(c,q) = D_{\text{JS}}(p_{\text{data}}, q)$. As shown in [HE16], we can generalize this construction by using different regularizers $\psi(c)$, and we end up with different divergence measures. Specifically, let $c(x) = \phi(-h(x))$, where $h$ is some function approximator with real-valued output. (The analysis above used $\phi(z) = \log \sigma(z)$ to arrive at the GAN objective). Let's define $\psi(c) = \mathbb{E}_{\text{data}}\left[-\phi(h(x)) - \phi(-h(x))\right]$.

$$\begin{aligned} L(c,q) &= \mathbb{E}_{\text{data}}\left[c(x)\right] - \mathbb{E}_q\left[c(x)\right] + H(q) + \psi(c) && (30)\\ &= \mathbb{E}_{\text{data}}\left[\phi(-h(x))\right] + \mathbb{E}_q\left[-\phi(-h(x))\right] + H(q) + \mathbb{E}_{\text{data}}\left[-\phi(h(x)) - \phi(-h(x))\right] && (31)\\ &= -\mathbb{E}_{\text{data}}\left[\phi(h(x))\right] - \mathbb{E}_q\left[\phi(-h(x))\right] + H(q) && (32) \end{aligned}$$

[NWJ09] show that when $h$ is allowed to range over the space of all functions, then the sum of expectations turns into an $f$-divergence:

$$\max_h \left( \mathbb{E}_{\text{data}}\left[\phi(h(x))\right] - \mathbb{E}_q\left[\phi(-h(x))\right] \right) = D_f(p_{\text{data}}, q) \quad (33)$$

$$\text{where} \quad f(u) = \max_h(-\phi(-h) - \phi(h)u) \quad (34)$$

Choosing $\phi(z) = \log \sigma(z)$ results in the Jensen-Shannon divegence, whereas other choices give different $f$-divergences; some possibilities are catalogued in [NWJ09].

### 6.2 $f$-GAN

Another approach for generalizing GANs and approximating $f$-divergences is in [NCT16]. That approach is more general than the one above using $\phi$-risks, as it allows one to approximate asymmetric $f$-divergences, however, the derivation of the Jensen-Shannon divergence involves a less natural set of choices.

## 7 Applications of GAN-like Methods

- **Inverse reinforcement learning**, as shown in [HE16]. Also [FLA16] frames their IRL approach using an energy-based model, and [Chr16] shows the close connections to GANs, including some interesting points about estimating the partition function using a mixture of $p_{\text{data}}$ and $q$, rather than $q$ alone.

- **Semi-supervised learning**, as shown in [Sal+16] and [Che+16].
- **Better unsupervised learning via lossy compression**. A natural method of formulating lossy compression in a universal way yields a minimax problem, as I described in my "Noise Should be Free" presentation. It may be possible to develop methods for density modeling that are better able to identify the interesting aspects of data using these ideas.
- **Model-based reinforcement learning**. Ask Jonathan Ho for details.
- **Sample-efficient reinforcement learning**. Ask Peter Chen for details.

# References


[Bub] Sebastian Bubeck. *ORF523 Course Notes*. https://blogs.princeton.edu/imabandit/2013/04/18/orf523-mirror-descent-part-iiii/.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[Che+16] Xi Chen et al. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: *arXiv preprint arXiv:1606.03657* (2016).

[Chr16] Paul Christiano. "Guided cost learning is generative adversarial modeling". In: *unpublished tech report* (2016).

[FLA16] Chelsea Finn, Sergey Levine, and Pieter Abbeel. "Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization". In: *arXiv preprint arXiv:1603.00448* (2016).

[Goo+14] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in Neural Information Processing Systems*. 2014, pp. 2672–2680.

[HE16] Jonathan Ho and Stefano Ermon. "Generative Adversarial Imitation Learning". In: *arXiv preprint arXiv:1606.03476* (2016).

[KB16] Taesup Kim and Yoshua Bengio. "Deep Directed Generative Models with Energy-Based Probability Estimation". In: *arXiv preprint arXiv:1606.03439* (2016).

[NCT16] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. "f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization". In: *arXiv preprint arXiv:1606.00709* (2016).

[NO09] Angelia Nedić and Asuman Ozdaglar. "Subgradient methods for saddle-point problems". In: *Journal of optimization theory and applications* 142.1 (2009), pp. 205–228.

[NWJ09] XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. "On surrogate loss functions and f-divergences". In: *The Annals of Statistics* (2009), pp. 876–904.

[RB16] Ernest K Ryu and Stephen Boyd. "Primer on monotone operator methods". In: *Appl. Comput. Math* 15.1 (2016), pp. 3–43.

[Sal+16] Tim Salimans et al. "Improved Techniques for Training GANs". In: *arXiv preprint arXiv:1606.03498* (2016).