Connecting Generative Adversarial Networks and Actor-Critic Methods

David Pfau, Oriol Vinyals
Google DeepMind
{pfau,vinyals}@google.com

Abstract

Both generative adversarial networks (GAN) in unsupervised learning and actor-critic methods in reinforcement learning (RL) have gained a reputation for being difficult to optimize. Practitioners in both fields have amassed a large number of strategies to mitigate these instabilities and improve training. Here we show that GANs can be viewed as actor-critic methods in an environment where the actor cannot affect the reward. We review the strategies for stabilizing training for each class of models, both those that generalize between the two and those that are particular to that model. We also review a number of extensions to GANs and RL algorithms with even more complicated information flow. We hope that by highlighting this formal connection we will encourage both GAN and RL communities to develop general, scalable, and stable algorithms for multilevel optimization with deep networks, and to draw inspiration across communities.

1 Introduction

Most problems in machine learning are formulated as an optimization problem over a single objective. However, a number of problems in machine learning lack a single unified cost, and instead consist of a hybrid of several models, each of which passes information to other models but tries to minimize its own private loss function. This upsets many of the assumptions behind most learning algorithms, and applying ordinary methods like gradient descent often leads to pathological behavior such as oscillations or collapse onto degenerate solutions. Despite these practical difficulties, there is great potential in models with hybrid or multilevel losses, and it has been hypothesized that the combination of many different local losses underlies the functioning of the brain as well [1].

Actor-critic methods (AC) [2, 3] and generative adversarial networks (GANs) [4] are two such classes of multilevel optimization problems which have close parallels. In both cases the information flow is a simple feedforward pass from one model which either takes an action (AC) or generates a sample (GANs) to a second model which evaluates the output of the first model. In both cases, the second model is the only one which has direct access to special information in the environment, either reward information (AC) or real samples from the distribution in question (GANs), and the first model must learn based on error signals from the second model alone. GANs and AC methods have important differences as well, and Section 2 we review both classes of algorithms and show a construction that bridges the two. Both of these models suffer from stability issues, and techniques for stabilizing training have been developed largely independently by the two communities. In Section 3 we review the different methods used to address model pathologies. We also provide a review of other related methods in the supplemental materials. The aim of this note is to raise awareness of the strong connection between the two classes of models.

(a) Generative Adversarial Networks [4]

(b) Deterministic Policy Gradient [7] / SVG(0) [8] / Neurally-Fitted Q-learning with Continuous Actions [9]
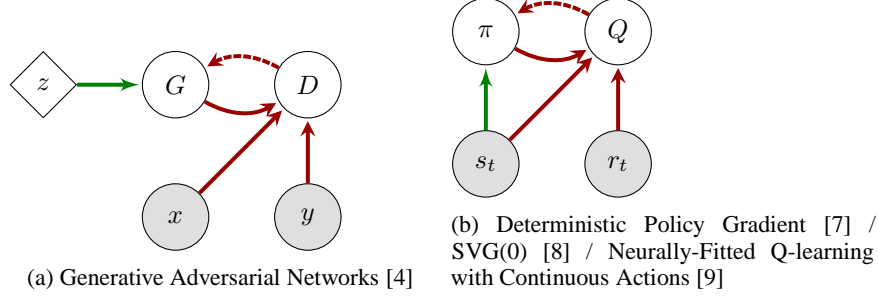
Figure 1: Information structure of GANs and AC methods. Empty circles represent models with a distinct loss function. Filled circles represent information from the environment. Diamonds represent fixed functions, both deterministic and stochastic. Solid lines represent the flow of information, while dotted lines represent the flow of gradients used by another model. Paths which are analogous between the two models are highlighted in red. The dependence of Q on future states and the dependence of future states on π are omitted for clarity.

2 Algorithms

Both GANs and AC can be seen as bilevel or two-time-scale optimization problems, where one model is optimized with respect to the optimum of another model:

$$x^* = \arg\min_{x \in \mathcal{X}} F(x, y^*(x)) \tag{1}$$

$$y^*(x) = \arg\min_{y \in \mathcal{Y}} f(x, y) \tag{2}$$

Bilevel optimization problems have been extensively studied in operations research, including the study of actor-critic methods [5], but largely in the setting where the two optimization problems are linear or convex programs [6]. By contrast, the recent applications we review here have largely used deep neural networks as the class of functions over which to optimize.

2.1 Generative Adversarial Networks

Generative adversarial networks [4] formulate the unsupervised learning problem as a game between two opponents - a generator G which samples from a distribution, and a discriminator D which classifies the samples as real or false. Typically the generator is represented as a deterministic feedforward neural network through which a fixed noise source $z \sim \mathcal{N}(0, I)$ is passed and the discriminator is another neural network which maps an image to a binary classification probability. The GAN game is then formulated as a zero sum game where the value is the cross-entropy loss between the discriminator's prediction and the true identity of the image as real or generated, which we denote y:

$$\min_{G} \max_{D} \mathbb{E}_{w,y}[y \mathrm{log} D(w) + (1 - y)\mathrm{log}(1 - D(w))] =$$
$$\min_{G} \max_{D} \mathbb{E}_{w \sim p_{\mathrm{data}}}[\mathrm{log} D(w)] + \mathbb{E}_{z \sim \mathcal{N}(0,I)}[\mathrm{log}(1 - D(G(z)))] \tag{3}$$

To make sure the generator has gradients from which to learn even when the discriminator's classification accuracy is high, the generator's loss function is usually formulated as maximizing the probability of classifying a sample as true rather than minimizing its probability of being classified false. The modified loss is still easily formulated as a bilevel optimization problem:

$$F(D, G) = -\mathbb{E}_{w \sim p_{\mathrm{data}}}[\mathrm{log} D(w)] + \mathbb{E}_{z \sim \mathcal{N}(0,I)}[\mathrm{log}(1 - D(G(z)))] \tag{4}$$

$$f(D, G) = -\mathbb{E}_{z \sim \mathcal{N}(0,I)}[\mathrm{log} D(G(z))] \tag{5}$$

2.2 Actor-Critic Methods

Actor-critic methods are a long-established class of techniques in reinforcement learning [2]. While most reinforcement learning algorithms either focus on learning a value function, like value iteration and TD-learning, or learning a policy directly, as in policy gradient methods, AC methods learn both simultaneously - the actor being the policy and the critic being the value function. In some AC methods, the critic provides a lower-variance baseline for policy gradient methods than estimating the value from returns. In this case even a bad estimate of the value function can be useful, as the policy gradient will be unbiased no matter what baseline is used. In other AC methods, the policy is updated with respect to the approximate value function, in which case pathologies similar to those in GANs can result. If the policy is optimized with respect to an incorrect value function, it may lead to a bad policy which never fully explores the space, preventing a good value function from being found and leading to degenerate solutions. A number of techniques exist to remedy this problem.

Formally, consider the typical MDP setting for RL, where we have a set of states $\mathcal{S}$, actions $\mathcal{A}$, a distribution over initial states $p_0(s)$, transition function $\mathcal{P}(s_{t+1}|s_t, a_t)$, reward distribution $\mathcal{R}(s_t)$ and discount factor $\gamma \in [0, 1]$. The aim of actor-critic methods is to simultaneously learn an action-value function $Q^\pi(s, a)$ that predicts the expected discounted reward:

$$Q^\pi(s, a) = \mathbb{E}_{s_{t+k} \sim \mathcal{P}, r_{t+k} \sim \mathcal{R}, a_{t+k} \sim \pi} \left[\sum_{k=1}^\infty \gamma^k r_{t+k} \middle| s_t = s, a_t = a \right] \tag{6}$$

and learn a policy that is optimal for that value function:

$$\pi^* = \arg\max_\pi \mathbb{E}_{s_0 \sim p_0, a_0 \sim \pi}[Q^\pi(s_0, a_0)] \tag{7}$$

We can express Q^π as the solution to a minimization problem:

$$Q^\pi = \arg\min_Q \mathbb{E}_{s_t, a_t \sim \pi}[\mathcal{D}(\mathbb{E}_{s_{t+1}, r_t, a_{t+1}}[r_t + \gamma Q(s_{t+1}, a_{t+1})] || Q(s_t, a_t))] \tag{8}$$

Where $\mathcal{D}(\cdot || \cdot)$ is any divergence that is positive except when the two are equal. Now the actor-critic problem can be expressed as a bilevel optimization problem as well:

$$
\begin{aligned}
F(Q, \pi) &= \mathbb{E}_{s_t, a_t \sim \pi}[\mathcal{D}(\mathbb{E}_{s_{t+1}, r_t, a_{t+1}}[r_t + \gamma Q(s_{t+1}, a_{t+1})] || Q(s_t, a_t))] & (9) \\
f(Q, \pi) &= -\mathbb{E}_{s_0 \sim p_0, a_0 \sim \pi}[Q^\pi(s_0, a_0)] & (10)
\end{aligned}
$$

There are many AC methods that attempt to solve this problem. Traditional AC methods optimize the policy through policy gradients and scale the policy gradient by the TD error, while the action-value function is updated by ordinary TD learning. We focus on deterministic policy gradients (DPG) [7, 10] and its extension to stochastic policies, SVG(0) [8], as well as neurally-fitted Q-learning with continuous actions (NFQCA) [9]. These algorithms are all intended for the case where actions and observations are continuous, and use neural networks for function approximation for both the action-value function and policy. This is an established approach in RL with continuous actions [11], and all methods update the policy by passing back gradients of the TD error with respect to the actions rather than passing the TD error directly. The distinction between the methods lies mainly in the way training proceeds. In NFQCA, the actor and critic are trained in batch mode after every episode, while in DPG and SVG(0), networks are trained online using temporal difference updates.

2.3 GANs as a kind of Actor-Critic

The similarities between GANs and AC methods are summarized in Figure 1. In both situations, one model has access to information about errors from the environment (the discriminator in GANs and the critic in AC), while the other model must be updated based only on gradient information from the first model.

We can make this connection more precise and describe an MDP in which GANs are a modified type of actor-critic method. Consider an MDP where the actions set every pixel in an image. The environment randomly chooses either to show the image the actor generates or show a real image.

Method	GANs	AC
Freezing learning	yes	yes
Label smoothing	yes	no
Historical averaging	yes	no
Minibatch discrimination	yes	no
Batch normalization	yes	yes
Target networks	n/a	yes
Replay buffers	no	yes
Entropy regularization	no	yes
Compatibility	no	yes

Table 1: Summary of different approaches used to stabilize and improve training for GANs and AC methods. Those approaches that have been shown to improve performance are in green, those that have not yet been demonstrated to improve training are in yellow, and those that are not applicable to the particular method are in red.

Let the reward from the environment be 1 if the environment chose the real image and 0 if not. This MDP is stateless as the image generated by the actor does not affect future data.

An actor-critic architecture learning in this environment clearly closely resembles the GAN game. A few minor adjustments have to be made to make it identical. If the actor had access to the state it could trivially pass a real image forward, so the actor must be a *blind* actor, with no knowledge of the state. Since the MDP is stateless this doesn't preclude the actor from performing well. While the mean-squared Bellman residual is usually used as a loss for the critic, cross-entropy should be used instead to match the GAN loss. This still gives a loss function with a consistent value function as the minimum. Lastly, the actor's parameters should not be updated if the environment shows a real image. This could be done by having the critic zero its gradients with respect to the action if the reward is 1. Thus GANs can be seen as a modified actor-critic methods with blind actors in stateless MDPs.

There are several peculiar aspects of this MDP that give rise to some behaviors not typically associated with actor-critic methods. First, it is not immediately obvious why an actor-critic algorithm should lead to adversarial behavior. Typically the actor and critic are trying to optimize complimentary loss functions, rather than optimize the same loss in different directions. The adversarial behavior in GANs comes about because the MDP in which the GAN game is played is one in which the actor cannot have any causal effect on the reward. Essentially, it is an MDP in which the true policy gradient is always zero. A critic, however, cannot learn the causal structure of the game from input examples alone, and moves in the direction of features that predict reward. The actor moves in a direction to increase reward based on the best estimate from the critic, but this change cannot lead to an increase in the true reward, so the critic will quickly learn to assign lower value in the direction the actor has moved. Thus the updates to the actor and critic, which ideally would be orthogonal (as in compatible actor-critic) instead becomes adversarial. It is also worth noting that there are important consequences to the environment being only partially observable to the actor. For MDPs, the optimal policy is always deterministic, but for GANs the desired generator is stochastic. If the actor did have access to the state, the optimal policy would be simple and deterministic - copy the environment. Thus it is critical that the actor be blind.

3   Stabilizing Strategies

Having reviewed the basics of GANs, actor-critic algorithms and their extensions, here we discuss the "tricks of the trade" used by each community to make them work. The different methods are summarized in Table 1. While not meant as an exhaustive list, we have included those that we believe have either made the largest impact in their fields or have the greatest potential for crossover between fields.

3.1 GANs

1. **Freezing learning**. Frequently either the generator or discriminator will outpace the other and the GAN will get stuck in a degenerate solution. One simple remedy is to freeze learning in one model when it begins to get too strong. A very similar approach has been taken successfully in actor-critic learning, where learning is frozen for either the actor or the critic if the magnitude of the TD error goes below or above a certain threshold [12].

2. **Label smoothing**. A simple trick to prevent gradients from vanishing when the discriminator's predictions are very confident, label smoothing replaces 0/1 labels with $\epsilon/1 - \epsilon$, which guarantees the generator will always have informative gradients. While specific to classification, there's no reason this couldn't equally well be applied in an RL setting where the rewards are 0/1 and gradients of the critic vanish.

3. **Historical averaging** [13]. Loosely inspired by fictitious play in game theory, historical averaging adds a drag term to gradient descent that penalizes steps that are too far from the past average over parameters. Even though the average over parameters doesn't have an intuitive meaning, this method is effective at preventing oscillations due to two models optimizing different objectives. Historical averaging is also closely related to Polyak-Ruppert averaging and extensions [14, 15]. While Polyak-Ruppert style averaging has been formally analyzed by those in the RL community [16], it has not been adopted as a standard "tool of the trade" to the best of our knowledge. The use of replay buffers in DPG [10] is also conceptually similar to fictitious play, but only applicable to the actor.

4. **Minibatch discrimination** [13]. To prevent the generator from collapsing onto a single sample, minibatch discrimination extends the role of the discriminator from classifying single images to classifying entire minibatches. This helps increase the entropy of samples from the generator. In RL, the analogous problem of underexploration has been addressed by adding a penalty to the actor that encourages higher entropy policies. Computing the entropy directly becomes impractical for complex stochastic policies, and an RL equivalent to minibatch discrimination could be an effective alternative for encouraging exploration in continuous spaces.

5. **Batch normalization** [17, 13]. Batch normalization has been critical for scaling GANs to deep convolutional networks, and virtual batch normalization has extended this by using a constant reference batch to prevent correlations in predictions due to the other elements of the minibatch. Batch normalization has also been helpful in a large fraction of environments DPG has been tried for, and it has not been investigated whether virtual batch normalization leads to even greater improvement.

3.2 Actor-Critic

1. **Replay buffers** [10]. Replay buffers have been very effective at removing correlations from the training data in both discrete and continuous RL settings. They are also conceptually similar to fictitious play in game theory, where one player plays the best response to the average over policies of the other player. Unlike fictitious play, where both players play against the historical average opponent, in the AC setting replay buffers can only be used for the critic. The critic can make off-policy updates based on actions chosen by the actor in the past, but the actor cannot learn from gradients from the critic in the past, since those gradients were taken with respect to different actions. Similarly for GANs, we can keep a buffer of previously generated images to prevent the discriminator from fitting too closely to the current generator. We have experimented with using a replay buffer for GANs, but were not able to generate asymptotically correct samples even for simple distributions.

2. **Target networks** [10]. Since the action value function appears twice in the Bellman recursion, stability can be an issue in Q-learning with function approximation. Target networks address this by fixing one of the networks in TD updates. Since the GAN game can be seen as a stateless MDP, the second appearance of the action value function disappears and learning the discriminator becomes an ordinary regression problem. For this reason we don't consider target networks applicable to the GAN setting. However other multilevel optimization problems with Q-learning as a subproblem would likely benefit. It's also worth noting that self-contrastive estimation [18] is conceptually similar to target networks, so the idea of a target network can still be applicable to other density estimation problems.

3. **Entropy regularization**. Actor-critic methods often fail to sufficiently explore the action space. To address this, an additional reward is sometimes added that encourages high-entropy policies. When the action space is discrete this is easily accomplished, but is somewhat more difficult in the continuous control setting. Notably, an analogous problem is often encountered with GANs, where the generator collapses onto sampling only a few modes. Any method for encouraging exploration in continuous control is likely transferable to increasing sample diversity in GANs.

4. **Compatibility**. One of the unique theoretical developments of actor-critic methods is the notion of a *compatible* critic. If the critic is a linear function of the gradient of the policy with respect to its parameters, then using this in place of the true value in the gradient of the expected return gives an unbiased approximation if the critic is optimal [2]. This compatible critic is also closely related to the natural gradient of the expected return, and can be used for efficient natural gradient descent on the policy [19]. While elegant, it is not clear if the notion of compatibility can be naturally extended to the GAN setting. Since the true value of any policy will always be 0.5 in the GAN MDP, the true policy gradient is always zero. We would generally prefer our GANs to be adversarial than compatible.

4 Conclusions

Combining deep learning with multilevel optimization holds great promise for a diverse array of problems in machine learning and AI. Already GANs and actor-critic methods have made large impacts on their respective fields, despite the inherent difficulties in optimization and exploration. We hope that by pointing out the deep connections between the two we encourage the development and adoption of general techniques and free flow of ideas between different communities.

References

[1] Adam Marblestone, Greg Wayne, and Konrad Kording. Towards an integration of deep learning and neuroscience. *Frontiers in Computational Neuroscience*, 10:94, 2016.

[2] Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 99, pages 1057–1063, 1999.

[3] Vijay R Konda and John N Tsitsiklis. On actor-critic algorithms. *SIAM journal on Control and Optimization*, 42(4):1143–1166, 2003.

[4] 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.

[5] VR Konda. *Actor-Critic Algorirhms*. PhD thesis, PhD Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2002.

[6] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153(1):235–256, 2007.

[7] David Silver, Guy Lever, Nicholas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

[8] Nicolas Heess, Gregory Wayne, David Silver, Tim Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pages 2944–2952, 2015.

[9] Roland Hafner and Martin Riedmiller. Reinforcement learning in feedback control. *Machine learning*, 84(1-2):137–169, 2011.

[10] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[11] Danil V Prokhorov and Donald C Wunsch. Adaptive critic designs. *IEEE transactions on Neural Networks*, 8(5):997–1007, 1997.

[12] Thomas Degris. personal communication.

[13] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

[14] Boris Teodorovich Polyak. A new method of stochastic approximation type. *Avtomatika i telemekhanika*, (7):98–107, 1990.

[15] David Ruppert. Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

[16] Vijay R Konda and John N Tsitsiklis. Convergence rate of linear two-time-scale stochastic approximation. *Annals of applied probability*, pages 796–819, 2004.

[17] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2016.

[18] Ian J Goodfellow. On distinguishability criteria for estimating generative models. *arXiv preprint arXiv:1412.6515*, 2014.

[19] Sham Kakade. A natural policy gradient. In *Advances in neural information processing systems*, volume 14, pages 1531–1538, 2001.

[20] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

[21] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

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

[23] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[24] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *arXiv preprint arXiv:1606.03476*, 2016.

[25] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. *arXiv preprint arXiv:1606.00709*, 2016.

[26] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial networks. *arXiv preprint arXiv:1609.03126*, 2016.

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

[28] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[29] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *arXiv preprint arXiv:1606.03657*, 2016.

[30] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[31] Pieter Abbeel and Andrew Y Ng. Exploration and apprenticeship learning in reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 1–8. ACM, 2005.

[32] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, pages 663–670, 2000.

[33] Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. In *Advances in neural information processing systems*, pages 1449–1456, 2007.

[34] John Schulman, Sergey Levine, Philipp Moritz, Michael I Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR, abs/1502.05477*, 2015.

Supplemental Material

(a) VAE/GAN hybrid [20]

(b) Bidirectional GAN [21] / Adversarially Learned Inference [22]

(c) Adversarial Autoencoder [23]

(d) SVG(1) [8]

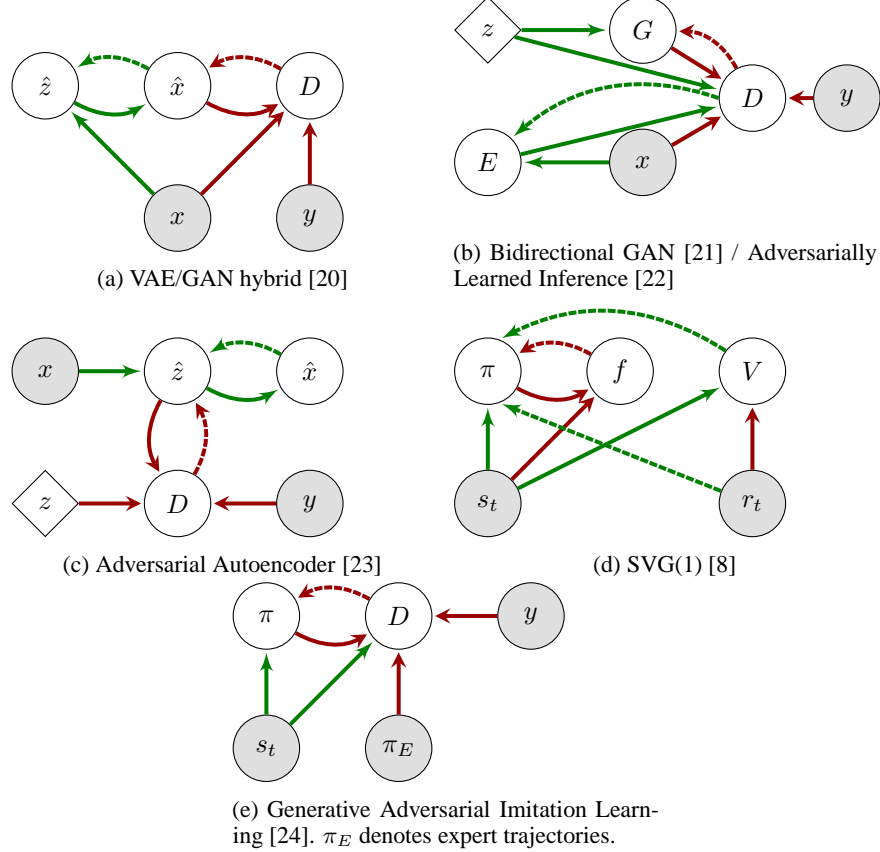(e) Generative Adversarial Imitation Learning [24]. π_E denotes expert trajectories.

Figure 2: Other machine learning problems which can be framed as multilevel optimization problems.

Beyond the basic GAN and AC methods described in Section 2, there have been many extensions and other problems in machine learning which have even greater complexity. We review some of them here.

A GAN extensions

There have been many extensions to GANs since the original publication. In [25] it was shown that GANs can be interpreted as minimizing a lower bound on the f-divergence between the generator distribution and true distribution:

$$\min_{G} \max_{T \in \mathcal{T}} \mathbb{E}_{w \sim p_{\text{data}}}[T(w)] - \mathbb{E}_{z \sim \mathcal{N}(0,I)}[f^*(T(G(z)))] \leq \min_{G} D_f(P_{\text{data}} || Q_G) \qquad (11)$$

Where Q_G is the distribution induced by the generator network G. For the appropriate choice of convex f the original GAN loss is recovered as a bound on the Jensen-Shannon divergence. Other choices of f lead to lower bounds on other divergences such as the KL divergence or squared Hellinger distance. GANs were also reformulated in [26] by replacing the discriminator with an energy-based autoencoder trained with a contrastive loss. Both f-GANs and energy-based GANs

preserve the information and gradient flow from the original GAN formulation but change the loss function.

GANs lack a natural way to infer the latent state for new data, leading to a number of extensions that include an inference network in addition to the generator and discriminator. A hybrid variational autoencoder [27, 28] and GAN architecture was trained in [20], while [22] and [21] both proposed an extension to GANs which trained the discriminator to classify the latent state of true data induced by an inference network in addition to the generated images. The structure of these models is illustrated in Figures 2a and 2b respectively. These extensions all added a third model to the mix, complicating optimization even further.

Further extensions include adversarial autoencoders [23] which adversarially learn the structure of the approximate posterior in variational autoencoders (shown in Figure 2c), and InfoGANs [29], which augment the latent state in the generative model with another set of units trained to maximize the mutual information with the data to learn the most important factors of variation in the model.

B   AC extensions

DPG, SVG(0) and NFQCA are not the only AC methods that work with continuous actions. Asynchronous advantage actor-critic (A3C) has shown promise for RL with both discrete and continuous actions [30]. A3C learns only a state value function $V(s_t)$ rather than an action value function $Q(s_t, a_t)$ and thus cannot pass back gradients of the value with respect to the action to the actor. Instead it approximates the action value with the rewards from several steps of experience and passes the TD error to the actor. It is therefore less closely connected to GANs, however it is worth pointing out that it has been a successful alternative to DPG and NFQCA as an actor-critic algorithm for continuous control.

Stochastic value gradients (SVG) [8] generalize DPG to stochastic policies in a number of ways, giving a spectrum from model-based to model-free algorithms. While SVG(0) is a direct stochastic generalization of DPG, SVG(1) combines an actor, critic and model f. The actor is trained through a combination of gradients from the critic, model and reward simultaneously.

C   Imitation Learning

Another machine learning problem closely related to GANs and actor-critic methods is that of learning control policies from expert trajectories, variously referred to as apprenticeship learning [31], imitation learning or inverse reinforcement learning [32] depending on the exact approach taken.

It has long been known that imitation learning can be formulated as a minimax problem where the aim is to learn the cost function that minimizes the amount by which the optimal policy outperforms the expert policy [33]. In [24], a deep connection was shown between imitation learning and GANs. Learning a cost function was shown to be equivalent to minimizing the regularized distance between state-action occupancy distributions for the expert policy and learned policy. By using a deep neural network with the cross-entropy loss as the distance between occupancy distributions, learning a cost directly could be circumvented and the inverse reinforcement learning problem was cast in nearly the same form as GAN learning. The only major difference from GANs is that instead of learning to generate images by gradient descent a policy is learned by trust region policy optimization [34]. In principle an actor-critic method could be substituted for direct policy optimization, and the imitation learning problem would become a three-level optimization problem that contains both GANs and actor-critic as subproblems.