OptionGAN: Learning Joint Reward-Policy Options using Generative Adversarial Inverse Reinforcement Learning

Peter Henderson[1], Wei-Di Chang[2], Pierre-Luc Bacon[1], David Meger[1], Joelle Pineau[1], Doina Precup[1]

[1] Department of Computer Science, McGill University, Montreal, Canada

[2] Department of Electrical, Computer, and Software Engineering, McGill University, Montreal, Canada

{peter.henderson,wei-di.chang}@mail.mcgill.ca
pbacon@cs.mcgill.ca
dmeger@cim.mcgill.ca
{jpineau,dprecup}@cs.mcgill.ca

Abstract

Reinforcement learning has shown promise in learning policies that can solve complex problems. However, manually specifying a good reward function can be difficult, especially for intricate tasks. Inverse reinforcement learning offers a useful paradigm to learn the underlying reward function directly from expert demonstrations. Yet in reality, the corpus of demonstrations may contain trajectories arising from a diverse set of underlying reward functions rather than a single one. Thus, in inverse reinforcement learning, it is useful to consider such a decomposition. The options framework in reinforcement learning is specifically designed to decompose policies in a similar light. We therefore extend the options framework and propose a method to simultaneously recover *reward options* in addition to policy options. We leverage adversarial methods to learn joint reward-policy options using only observed expert states. We show that this approach works well in both simple and complex continuous control tasks and shows significant performance increases in one-shot transfer learning.

Introduction

A long term goal of Inverse Reinforcement Learning (IRL) is to be able to learn underlying reward functions and policies solely from human video demonstrations. We call such a case, where the demonstrations come from different contexts and the task must be performed in a novel environment, one-shot transfer learning. However, such demonstrations would undoubtedly come from a wide range of settings and environments. For example, given only demonstrations of a human walking on earth, can an agent learn to walk on the moon?

In this case, expert demonstrations may not conform to a single reward function. This proves detrimental to current methods which might over-generalize and cause poor performance. In forward RL, decomposing a policy into smaller specialized policy options has been shown to improve results for exactly such cases (Sutton, Precup, and Singh 1999; Bacon, Harb, and Precup 2016). Thus, we extend the options framework to IRL and decompose both the reward function and policy. Our method is able to learn deep policies which can specialize to the set of best-fitting experts. Hence, it excels at one-shot transfer learning where single-approximator methods waver.

To accomplish this, we make use of the Generative Adversarial Imitation Learning (GAIL) (Ho and Ermon 2016) framework and formulate a method for learning joint reward-policy options with adversarial methods in IRL. As such, we call our method OptionGAN. This method can implicitly learn divisions in the demonstration state space and accordingly learn policy and reward options. Leveraging a correspondence between Mixture-of-Experts (MoE) and one-step options, we learn a decomposition of rewards and the policy-over-options in an end-to-end fashion. This decomposition is able to capture simple problems and learn any of the underlying rewards in one shot. This gives flexibility and benefits for a variety of future applications (both in reinforcement learning and standard machine learning).

We evaluate OptionGAN in the context of continuous control locomotion tasks, considering both simulated MuJoCo locomotion OpenAI Gym environments (Brockman et al. 2016), modifications of these environments for task transfer (Henderson et al. 2017), and a more complex Roboschool task (Schulman et al. 2017). We show that the final policies learned using joint reward-policy options outperform a single reward approximator and policy network in most cases, and particularly excel at one-shot transfer learning.

Related Work

One goal in robotics research is to create a system which learns how to accomplish complex tasks simply from observing an expert's actions (such as videos of humans performing actions). While IRL has been instrumental in working towards this goal, it has become clear that fitting a single reward function which generalizes across many domains is difficult. To this end, several works investigate decomposing the underlying reward functions of expert demonstrations for IRL (Krishnan et al. 2016; Sermanet, Xu, and Levine 2016). In (Krishnan et al. 2016), reward functions are decomposed into a set of subtasks based on segmenting expert demonstration transitions (known state-action pairs) by analyzing the changes in "local linearity with respect to a kernel function". Similarly, in (Sermanet, Xu, and Levine 2016), techniques in video editing based on information-similarity are adopted to divide a video demonstration into distinct sections which can then be recombined into a differentiable reward function.

However, simply decomposing the reward function may

not be enough, the policy must also be able to adapt to different tasks. Several works have investigated learning a latent structure for such a purpose (Merel et al. 2017; Hausman et al. 2017; Wang et al. 2017). These methods attempt to use a latent variable to elicit decomposed behaviours from multiple tasks learned at once. In (Merel et al. 2017) the authors use generative adversarial imitation learning to learn several low-level controllers from single task demonstrations based on motion-capture data. These low-level policies are then combined together with a high-level policy to bootstrap the learning of a forward RL task. In (Hausman et al. 2017; Wang et al. 2017), a latent dimension is learned in a similar manner to InfoGAN (Chen et al. 2016) along with the desired policy. This latent dimension allows multiple tasks to be learned by one policy and elicited by subsequently passing a latent variable to the policy.

In contrast, our work focuses on one-shot transfer learning. In the former work, the desired latent variable must be known and provided, whereas in our formulation the latent structure is inherently encoded in an unsupervised manner. This is inherently accomplished while learning to solve a task composed of a wide range of underlying reward functions and policies in a single framework.

Preliminaries and Notation

Markov Decision Processes (MDPs) MDPs consist of states S, actions A, a transition function $P : S \times A \to (S \to \mathbb{R})$, and a reward function $r : S \to \mathbb{R}$. We formulate our methods in the space of continuous control tasks ($A \in \mathbb{R}, S \in \mathbb{R}$) using measure-theoretic assumptions. Thus we define a parameterized *policy* as the probability distribution over actions conditioned on states $\pi_\theta : S \times A \to [0, 1]$, modeled by a Gaussian $\pi_\theta \sim \mathcal{N}(\mu, \sigma^2)$ where θ are the policy parameters. The value of a policy is defined as $V_\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t r_{t+1} | s_0 = s]$ and the action-value is $Q_\pi(s, a) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t r_{t+1} | s_0 = s, a_0 = a]$, where $\gamma \in [0, 1)$ is the discount factor.

The Options framework In reinforcement learning, an option ($\omega \in \Omega$) can be defined by a triplet $(I_\omega, \pi_\omega, \beta_\omega)$. In this definition, π_ω is called an intra-policy option, $I_\omega \subseteq S$ is an initiation set, and $\beta_\omega : S \to [0, 1]$ is a termination function (that is the probability that an option ends at a given state) (Sutton, Precup, and Singh 1999). Furthermore, π_Ω is the policy-over-options. That is, π_Ω determines which option π_ω an agent picks to use until the termination function β_ω indicates that a new option should be chosen. Other works explicitly formulate *call-and-return* options, but we instead simplify to *one-step* options, where $\beta_\omega(s) = 1; \forall \omega \in \Omega, \forall s \in S$. One-step options have long been discussed as an alternative to temporally extended methods and often provide advantages in terms of optimality and value estimation (Sutton, Precup, and Singh 1999; Dietterich 1998). Furthermore, we find that our options still converge to temporally extended and interpretable actions.

Mixture-of-Experts The idea of creating a mixture of experts (MoEs) was initially formalized to improve learning of neural networks by dividing the input space among several networks and then combining their outputs through a soft weighted average (Jacobs et al. 1991). It has since come into prevalence for generating extremely large neural networks (Shazeer et al. 2017). In our formulation of joint reward-policy options, we leverage a correspondence between Mixture-of-Experts and options. In the case of one-step options, the policy-over-options (π_Ω) can be viewed as a deterministic specialized gating function over experts (intra-options policies $\pi_\omega(s)$): $\sum_\omega \pi_\Omega(\omega, s)\pi_\omega(s)$. Several works investigate convergence to a sparse and specialized Mixture-of-Experts (Jacobs et al. 1991; Shazeer et al. 2017). We leverage these works to formulate a Mixture-of-Experts which converges to one-step options.

Policy Gradients Policy gradient (PG) methods (Sutton et al. 2000) formulate a method for optimizing a parameterized policy π_θ through stochastic gradient descent. In the discounted setting, PG methods optimize $\rho(\theta, s_0) = \mathbb{E}_{\pi_\theta}[\sum_{t=0}^\infty \gamma^t r(s_t)]$. The policy gradient theorem formulates: $\frac{\delta\rho(\theta, s_0)}{\delta\theta} = \sum_s \mu_{\pi_\theta}(s|s_0)\sum_a \frac{\delta\pi_\theta(a|s)}{\delta\theta}Q_{\pi_\theta}(s, a)$, where $\mu_{\pi_\theta}(s|s_0) = \sum_{t=0}^\infty \gamma^t P(s_t = s|s_0)$ (Sutton et al. 2000). In Trust Region Policy Optimization (TRPO) (Schulman et al. 2015a) and Proximal Policy Optimization (PPO) (Schulman et al. 2017) this update is constrained and transformed into the advantage estimation view. That is, the above becomes a constrained optimization: $\max_\theta \mathbb{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}A_t\right]$ subject to $\mathbb{E}_t\left[\text{KL}\left[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)\right]\right] \leq \delta$ where A_t is the generalized advantage function (Schulman et al. 2015b). In TRPO, this is solved as a constrained conjugate gradient descent problem, while in PPO the constraint is transformed into a penalty term (or reformulated as a clipping objective).

Inverse Reinforcement Learning Inverse Reinforcement Learning was first formulated in the context of Markov Decision Processes (MDPs) by (Ng and Russell 2000). In later work, a parametrization of the reward function is learned as a linear combination of the state feature expectation so that the hyperdistance between the expert and the novice's feature expectation is minimized (Abbeel and Ng 2004). It has also been shown that a solution can be formulated using the maximum entropy principle, with the goal of matching feature expectation as well (Ziebart et al. 2008). Generative adversarial imitation learning (GAIL) make use of adversarial techniques to perform a similar feature expectation matching (Ho and Ermon 2016). In this case, a discriminator uses state-action pairs (transitions) from the expert demonstrations and novice rollouts to learn a binary classification probability distribution. The probability that a state belongs to an expert demonstration can then be used as the reward for a policy optimization step. However, unlike GAIL, we do not assume knowledge of the expert actions. Rather, we rely solely on observations in the discriminator problem. We therefore refer to our baseline approach as Generative Adversarial Inverse Reinforcement Learning (IRLGAN) as opposed to imitation learning. As such, our adversarial game optimizes:

$$\max_{\pi_\Theta} \min_{R_{\hat{\Theta}}} -\left[\mathbb{E}_{\pi_\Theta}[\log R_{\hat{\Theta}}(s)] + \mathbb{E}_{\pi_E}[\log(1 - R_{\hat{\Theta}}(s))]\right]$$

(1)

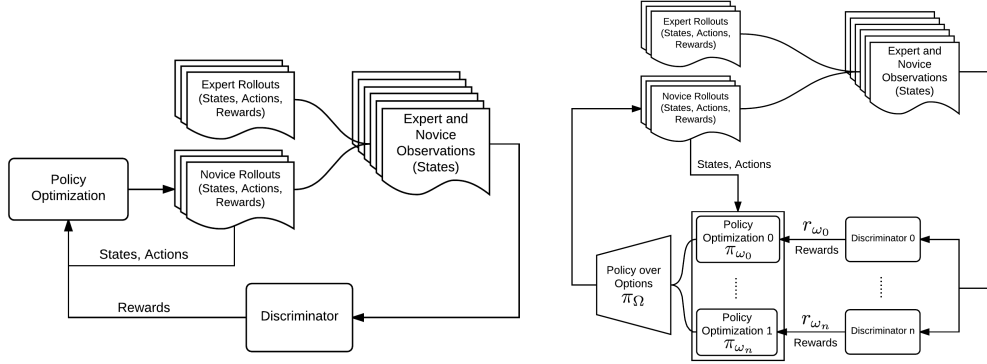where π_Θ and π_E are the policy of the novice and expert

Figure 1: Generative Adversarial Inverse Reinforcement Learning (left) and OptionGAN (right) Architectures

parameterized by Θ and E, respectively, and $R_{\hat{\Theta}}$ is the discriminator probability that a sample state belongs to an expert demonstration (parameterized by $\hat{\Theta}$). In our case the discriminator approximates a reward function, and as such we use the notion of $R_{\hat{\Theta}}$ to denote the discriminator. Similarly to GAIL, we use TRPO during the policy optimization step for simple tasks. However, for complex tasks we adopt PPO. Figure 1 and Algorithm 1 show an outline for the general IRLGAN process.

Algorithm 1: IRLGAN

Input : Expert trajectories $\tau_E \sim \pi_E$.

1 Initialize $\Theta, \hat{\Theta}$
2 **for** $i = 0, 1, 2, \ldots$ **do**
3   Sample trajectories $\tau_N \sim \pi_{\Theta_i}$
4   Update discriminator parameters ($\hat{\Theta}$) according to:

$$L_{\hat{\Theta}} = \mathbb{E}_{s \sim \tau_N}[\log(R_{\hat{\Theta}}(s))] + \mathbb{E}_{s \sim \tau_E}[\log(1 - R_{\hat{\Theta}}(s))]$$

5   Update policy (with constrained update step and parameters θ) according to:

$$\mathbb{E}_{\tau_N}[\nabla_{\Theta} \log \pi_{\Theta_i}(a|s) \mathbb{E}_{\tau_N}[\log(R_{\hat{\Theta}_{i+1}}(s))|s_0 = \bar{s}]]$$

6 **end**

Reward-Policy Options Framework

Based on the need to infer a decomposition of underlying reward functions from a wide range of expert demonstrations in one-shot transfer learning, we can extend the options framework for decomposing rewards as well as policies. In this way, intra-option policies, decomposed rewards, and the policy-over-options can all be learned in concert in a cohesive framework. In this case, an option can be formulated by a tuple: $(I_\omega, \pi_\omega, \beta_\omega, r_\omega)$. Here, r_ω is a reward option which matches a policy option π_ω. The policy-over-options not only chooses the intra-option policy, but the reward option as well: $\pi_\Omega \rightarrow (r_\omega, \pi_\omega)$. For simplicity, we refer to the policy over reward options as r_Ω (though in our formulation it is desirable to set $r_\Omega = \pi_\Omega$).

To learn joint reward-policy options, we present a method which fits into the framework of IRLGAN. We reformulate the discriminator as a Mixture-Of-Experts and re-use the gating function when learning a set of policy options. We show that by properly formulating the discriminator loss function, the Mixture-Of-Experts converges to one-step options. This formulation also allows us to use regularizers which encourage distribution of information, diversity, and sparsity in both the reward and policy options.

Learning Joint Reward-Policy Options

The use of one-step options allows us to learn a policy-over-options in an end-to-end fashion as a Mixture-of-Experts formulation. In the one-step case, selecting an option $(\pi_{\omega,\theta})$ using the policy-over-options $(\pi_{\Omega,\zeta})$ can be viewed as a mixture of completely specialized experts: $\pi_\Theta(a|s) = \sum_\omega \pi_{\Omega,\zeta}(\omega, s) \pi_{\omega,\theta}(s)$. Similarly, the reward for a given state can be viewed as: $R_{\Omega,\hat{\Theta}}(s) = \sum_\omega \pi_{\Omega,\zeta}(\omega, s) r_{\omega,\hat{\theta}}(s)$. Here, $\zeta, \theta \in \Theta, \hat{\theta} \in \hat{\Theta}$ are the parameters of the policy-over-options, policy options, and reward options, respectively.

As such, we reformulate our discriminator loss as a weighted mixture of completely specialized experts in Eq. 10. This allows us to update the parameters of the policy-over-options and our reward options together as part of the discriminator step.

$$L_\Omega = \mathbb{E}_\omega \left[\pi_{\Omega,\zeta}(\omega, s) L_{\hat{\theta},\omega} \right] + L_{reg} \qquad (2)$$

Here, $L_{\hat{\theta},\omega}$ is the sigmoid cross-entropy loss of the reward options (discriminators) and L_{reg} is any penalty which can encourage certain properties of the policy-over-options or the overall reward signal. As can be seen in Algorithm 2 and Figure 1, this loss function can fit directly into the IRLGAN framework.

Having updated the parameters of the policy-over-options and our reward options, standard PG methods can be used to optimize the parameters of our intra-option policies. This can be done by weighting the average of the intra-option policy actions with the policy-over-options $\pi_{\Omega,\zeta}$. While it is possible to update each intra-option policy separately as in (Bacon, Harb, and Precup 2016), this Mixture-of-Experts

---
Algorithm 2: OptionGAN

Input : Expert trajectories $\tau_E \sim \pi_E$.

1 Initialize $\theta, \hat{\theta}$

2 **for** $i = 0, 1, 2, \ldots$ **do**

3 $\quad$ Sample trajectories $\tau_N \sim \pi_{\Theta_i}$

4 $\quad$ Update discriminator options parameters $\hat{\theta}, \omega$ and policy-over-options parameters ζ, to minimize:

$$L_{\hat{\theta},\omega} = \mathbb{E}_{s \sim \tau_N}[\log(r_{\hat{\theta},\omega}(s))] + \mathbb{E}_{s \sim \tau_E}[\log(1 - r_{\hat{\theta},\omega}(s))]$$

$$L_\Omega = \mathbb{E}_\omega \left[\pi_{\Omega,\zeta}(\omega, s) L_{\hat{\theta},\omega} \right] + L_{reg}$$

5 $\quad$ Update policy options (with constrained update step and parameters $\theta_\omega \in \Theta_\Omega$) according to:

$$\mathbb{E}_{\tau_N}[\nabla_\theta \log \pi_\Theta(a|s)\mathbb{E}_{\tau_N}[\log(R_{\Omega,\hat{\Theta}}(s))|s_0 = \bar{s}]]$$

6 **end**
---

formulation is equivalent, as discussed in the next section. Once the gating function specializes over the options, all gradients except for those related to the intra-option policy selected would be weighted by zero. We find that this end-to-end parameter update formulation leads to easier implementation and smoother learning with constraint-based methods.

Mixture-of-Experts as Options

To ensure that our Mixture-Of-Experts formulation converges to options in the optimal case, we must properly formulate our loss function such that the gating function specializes over experts. While it may be possible to force a sparse selection of options through a top-k choice as in (Shazeer et al. 2017), we find that this leads to instability since for $k = 1$ the top-k function is not differentiable. As is specified in (Jacobs et al. 1991), a loss function of the form $L = (y - \frac{1}{||\Omega||}\sum_\omega \pi_\Omega(\omega, s)y_\omega(s))^2$ draws cooperation between experts, but a reformulated loss function of the form $L = \frac{1}{||\Omega||}\sum_\omega \pi_\Omega(\omega, s)(y - y_\omega(s))^2$ encourages specialization.

If we view our policy-over-options as a softmax $\pi_\Omega(\omega, s) = \frac{\exp(z_\omega(s))}{\sum_i \exp(z_i(s))}$, then the derivative of the loss function with respect to the gating function becomes:

$$\frac{dL}{dz_\omega} = \frac{1}{||\Omega||}\pi_\Omega(\omega, s)\left((y - y_\omega(s))^2 - L\right) \quad (3)$$

This can intuitively be interpreted as encouraging the gating function to increase the likelihood of choosing an expert when its loss is less than the average loss of all the experts. The gating function will thus move toward deterministic selection of experts.

As we can see in Eq. 10, we formulate our discriminator loss in the same way, using each reward option and the policy-over-options as the experts and gating function respectively. This ensures that the policy-over-options specializes over the state space and converges to a deterministic selection of experts. Hence, we can assume that in the optimal

case, our formulation of an MoE-style policy-over-options is equivalent to one-step options. Our characterization of this notion of Mixture-of-Experts as options is further backed by experimental results. Empirically, we still find temporal coherence across option activation despite not explicitly formulating call-and-return options as in (Bacon, Harb, and Precup 2016).

Regularization Penalties

Due to our formulation of Mixture-of-Experts as options, we can learn our policy-over-options in an end-to-end manner. This allows us to add additional terms to our loss function to encourage the appearance of certain target properties.

Sparsity and Variance Regularization To ensure an even distribution of activation across the options, we look to conditional computation techniques that encourage sparsity and diversity in hidden layer activations and apply these to our policy-over-options (Bengio et al. 2015). We borrow three penalty terms L_b, L_e, L_v (adopting a similar notation). In the minibatch setting, these are formulated as:

$$L_b = \sum_\omega ||\mathbb{E}_s[\pi_\Omega(\omega, s)] - \tau||_2 \quad (4)$$

$$L_e = \mathbb{E}_s \left[|| \left(\frac{1}{||\Omega||}\sum_\omega \pi_\Omega(\omega, s) \right) - \tau||_2 \right] \quad (5)$$

$$L_v = -\sum_\omega var_\omega\{\pi_\Omega(s)\} \quad (6)$$

where τ is the target sparsity rate (which we set to .5 for all cases). Here, L_b encourages the activation of the policy-over-options with target sparsity τ "in expectation over the data" (Bengio et al. 2015). Essentially, L_b encourages a uniform distribution of options over the data while L_e drives toward a target sparsity of activations per example (doubly encouraging our mixtures to be sparse). L_v also encourages varied π_Ω activations while discouraging uniform selection.

Mutual Information Penalty To ensure the specialization of each option to a specific partition of the state space, a mutual information (MI) penalty is added[1]. We thus minimize mutual information pairwise between option distributions, similarly to (Liu and Yao 2002):

$$I(F_i; F_j) = -\frac{1}{2}\log(1 - \rho_{ij}^2), \quad (7)$$

where F_i and F_j are the outputs of reward options i and j respectively, and ρ_{ij} the correlation coefficient of F_i and F_j, defined as $\rho_{ij} = \frac{E[(F_i - E[F_i])(F_j - E[F_j])]}{\sigma_i^2 \sigma_j^2}$.

The resulting loss term is thus computed as:

$$L_{\text{MI}} = \sum_{\omega \in \Omega}\sum_{\hat{\omega} \in \Omega, \omega \neq \hat{\omega}} I(\pi_\omega, \pi_{\hat{\omega}}). \quad (8)$$

---

[1] While it may be simpler to use an entropy regularizer, we found that in practice it performs worse. Entropy regularization encourages exploration as is highlighted in (Mnih et al. 2016). In the OptionGAN setting, this ends up making convergence unstable, while the mutual information term cooperatively encourages diversity in the options while providing stable learning.
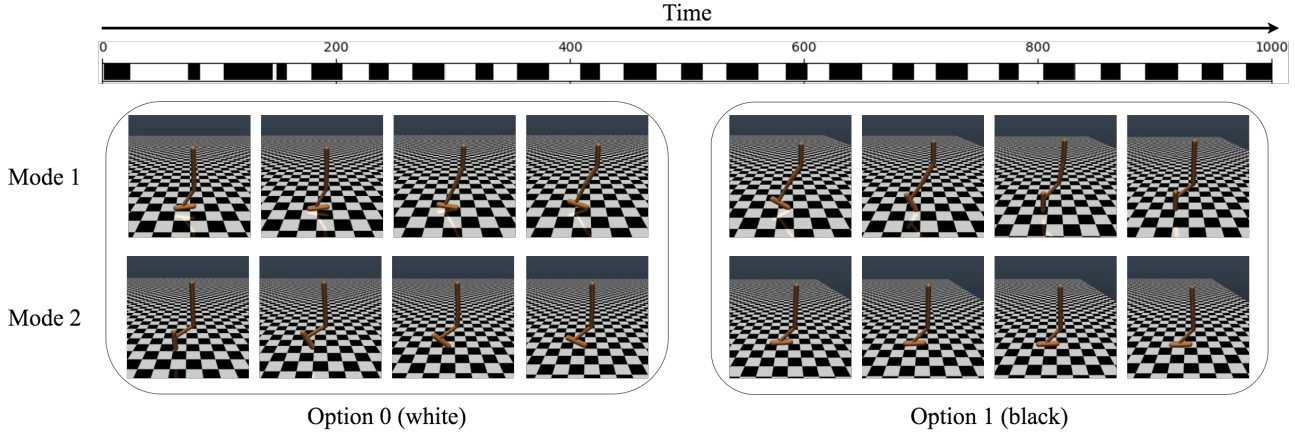
Figure 2: The policy-over-options elicits two interpretable behaviour modes per option, but temporal cohesion and specialization is seen between these behaviour modes across time within a sample rollout trajectory.

Thus the overall regularization term becomes:

$$\mathcal{L}_{reg} = \lambda_b L_b + \lambda_e L_e + \lambda_v L_v + \lambda_{\text{MI}} L_{\text{MI}}. \qquad (9)$$

Experiments

To evaluate our method of learning joint reward-policy options, we investigate continuous control tasks. We divide our experiments into 3 settings: simple locomotion tasks, one-shot transfer learning from a mixture of demonstrations to an unseen environment, and complex tasks.

We compare OptionGAN against IRLGAN in all scenarios, investigating whether dividing the reward and policy functions into options improves performance against the single approximator case. All shared hyperparameters are constant between the IRLGAN and OptionGAN evaluation runs[2]. All evaluations are averaged across 10 trials, each using a different random seed. For all experiments, the average return of the true reward function across 25 sample rollouts is used as the target evaluation metric. Table 1 shows the overall results of our evaluations. Here, we highlight certain learning curves, with extended details and all learning curves provided in the supplemental. We find that in nearly every setting, the final optionated policy learned by Option-GAN outperforms the single approximator case when keeping all other shared hyperparameters equal (e.g. learning rates, network architectures).

Network Architecture and Essential Hyperparameters

We use multilayer perceptrons (MLPs) for both the value and policy networks as in (Ho and Ermon 2016). However, we use shared layers for our OptionGAN intra-option policy and reward networks: $r_\omega, \forall \omega \in \Omega$ all share hidden layers and $\pi_\omega, \forall \omega \in \Omega$ share hidden layers. We use separate parameters

---

[2]Full experimental details can be found in the supplemental material, with code provided at https://github.com/Breakend/OptionGAN.

for the policy-over-options π_Ω as we share them between the reward and policy options. Shared layers are used to ensure a fair comparison against a single network of the same number of hidden layers in both the learned reward function and policy. For simple settings all hidden layers are of size $(64, 64)$ and for complex experiments are $(128, 128)$. For the 2-options case we set $\lambda_e = 10.0, \lambda_b = 10.0, \lambda_v = 1.0$ based on a simple hyperparameter search and reported results from (Bengio et al. 2015). For the 4-options case we relax the regularizer encouraging a uniform distribution of options (L_b), setting $\lambda_b = .01$.

Simple Tasks

First, we investigate simple settings with no transfer for a set of benchmark locomotion tasks provided in OpenAI Gym (Brockman et al. 2016) using the MuJoCo (Todorov, Erez, and Tassa 2012) simulator. We use the Hopper-v1, HalfCheetah-v1, and Walker2d-v1 locomotion environments. The results of this experiment are shown in Table 1 and sample learning curves for Hopper and HalfCheetah can be found in Figure 6. We use 10 expert rollouts from a policy trained using TRPO for 500 iterations.

In these simple settings, OptionGAN converges to policies which perform as well or better than the single approximator setting. Importantly, even in these simple settings, the options which our policy selects have a notion of temporal coherence and interpretability despite not explicitly enforcing this in the form of a termination function. This can be seen in the two option version of the Hopper-v1 task in Figure 2. We find that generally each option takes on two behaviour modes. The first option handles: (1) the rolling of the foot during hopper landing; (2) the folding in of the foot in preparation for floating, while the second option handles: (1) the last part of take-off where the foot is hyper-extended and the body flexed; (2) the part of air travel where movement is locked.

Task	Expert	IRLGAN	OptionGAN (2ops)	OptionGAN (4ops)
Hopper-v1	3778.8 ± 0.3	**3736.3 ± 152.4**	*3641.2 ± 105.9*	*3715.5 ± 17.6*
HalfCheetah-v1	4156.9 ± 8.7	3212.9 ± 69.9	**3714.7 ± 87.5**	3616.1 ± 127.3
Walker2d-v1	5528.5 ± 7.3	*4158.7 ± 247.3*	*3858.5 ± 504.9*	**4239.3 ± 314.2**
Hopper (One-Shot)	3657.7 ± 25.4	2775.1 ± 203.3	3409.4 ± 80.8	**3464.0 ± 67.8**
HalfCheetah (One-Shot)	4156.9 ± 51.3	1296.3 ± 177.8	1679.0 ± 284.2	**2219.4 ± 231.8**
Walker (One-Shot)	4218.1 ± 43.1	3229.8 ± 145.3	**3925.3 ± 138.9**	3769.40 ± 170.4
HopperSimpleWall-v0	3218.2 ± 315.7	*2897.5 ± 753.5*	*3140.3 ± 674.3*	**3272.3 ± 569.0**
RoboschoolHumanoidFlagrun-v1	2822.1 ± 531.1	*1455.2 ± 567.6*	*1868.9 ± 723.7*	**2113.6 ± 862.9**

Table 1: True Average Return with the standard error across 10 trials on the 25 final evaluation rollouts using the final policy.

One-Shot Learning from Mixed Expert Demonstrations

We also investigate one-shot transfer learning. In this scenario, the novice is trained on a target environment, while expert demonstrations come from a similar task, but from different environments with altered dynamics (one-shot transfer from varied expert demonstrations to a new environment). To demonstrate the effectiveness of OptionGAN in these settings, we use expert demonstrations from environments with varying gravity conditions as seen in (Henderson et al. 2017; Christiano et al. 2016). We vary the gravity (.5, .75, 1.25, 1.5 of Earth's gravity) and train experts using TRPO for each of these. We gather 10 expert trajectories from each gravity variation for a total of 40 expert rollouts. We then use the observations to train an agent on the normal Earth gravity environment (the default -v1 environment as provided in OpenAI Gym). We do this for Hopper-v1, HalfCheetah-v1, and Walker2D-v1. We select these gravity tasks due to the demonstration in (Henderson et al. 2017) that learning sequentially on these varied gravity environments causes catastrophic forgetting of the policy on environments seen earlier in training. This suggests that the dynamics are varied enough that trajectories are difficult to generalize across, but still share some state representations and task goals. As seen in Figure 6, using options can cause significant performance increases in this area, but performance gains can vary across the number of options and the regularization penalty as seen in Table 1.

Complex Tasks

Lastly, we investigate slightly more complex tasks. We use the HopperSimpleWall-v0 environment from the gym-extensions framework (Henderson et al. 2017) and the RoboschoolHumanoidFlagrun-v1 environment (Schulman et al. 2017). In the first, a wall is placed randomly in the path of the Hopper-v1 agent and simplified sensor readouts are added to the observations as in (Wang et al. 2017). In the latter, the goal is to run and reach a frequently changing target. This is an especially complex task with a highly varied state space. In both cases we use an expert trained with TRPO and PPO respectively, to generate 40 expert rollouts. For the Roboschool environment, we find that TRPO does not allow enough exploration to perform adequately, and thus we switch our policy optimization method to the clipping-objective version of PPO.

Ablation Investigations

Convergence of Mixtures to Options To show that our formulation of Mixture-of-Experts decomposes to options in the optimal case, we investigate the distributions of our policy-over-options. We find that across 40 trials, 100% of activations fell within a reasonable error bound of deterministic selection across 1M samples. That is, in 40 total trials across 4 environments (Hopper-v1, HalfCheetah-v1, Walker2d-v1, RoboschoolHumanoidFlagrun-v1), policies were trained for 500 iterations (or 5k iterations in the case of RoboschoolHumanoidFlagrun-v1). We collected 25k samples at the end of each trial. Among the gating activations across the samples, we recorded the number of gating activations within the range $\{0 + \epsilon, 1 - \epsilon\}$ for $\epsilon = 0.1$. 100% fell within this range. 98.72% fell within range $\epsilon = 1^{-3}$. Thus at convergence, both intuitively and empirically we can refer to our gating function over experts as the policy-over-options and each of the experts as options.

Effect of Uniform Distribution Regularizer We find that forcing a uniform distribution over options can potentially be harmful. This can be seen in the experiment in Figure 4, where we evaluate the 4 option case with $\lambda_b = \{0.1, 10\}$. However, relaxing the uniform constraint results in rapid performance increases, particularly in the Half-Cheetah environment where we see dramatic increases in learning speed and performance with 4 options.

There is an intuitive explanation for this. In the 4-option case, with a relaxed uniform distribution penalty, we allow options to drop out during training. In the case of Hopper and Walker tasks, generally 2 options drop out slowly over time, but in Half-Cheetah, only one option drops out in the first 20 iterations with a uniform distribution remaining across the remaining options as seen in Figure 6. We posit that in the case of Half-Cheetah there is enough mutually exclusive information in the environment state space to divide across 3 options, quickly causing a rapid gain in performance, while the Hopper tasks do not settle as quickly and thus do not see that large gain in performance.

Latent Structure in Expert Demonstrations Another benefit of using options in the IRL transfer setting is that the underlying latent division of the original expert environments is learned by the policy-over-options. As seen in Figure 9, the expert demonstrations have a clear separation among options. We suspect that options further away from the target gravity are not as specialized due to the fact that their state spaces are covered significantly by a mixture of
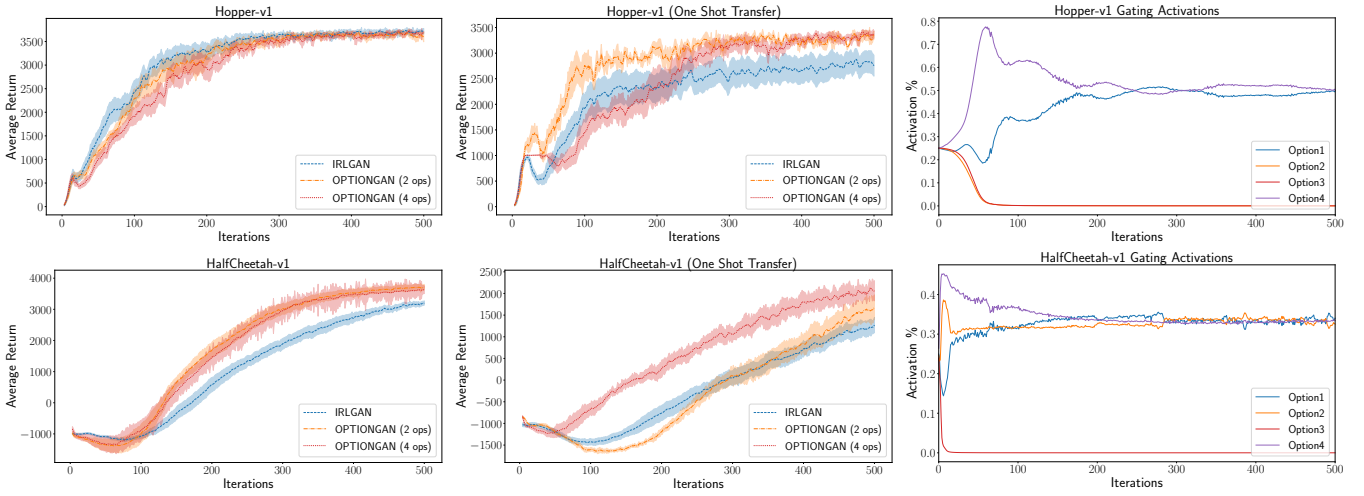
Figure 3: Left Column: Simple locomotion curves. Error bars indicate standard error of average returns across 10 trials on 25 evaluation rollouts. Middle Column: One-shot transfer experiments with 40 expert demonstrations from varied gravity environments without any demonstrations on the novice training environment training on demonstrations from .5G, .75G, 1.25G, 1.5G gravity variations. Right Column: Activations of policy-over-options over time with 4 options on training samples in the one-shot transfer setting with $\lambda_b = .01$.
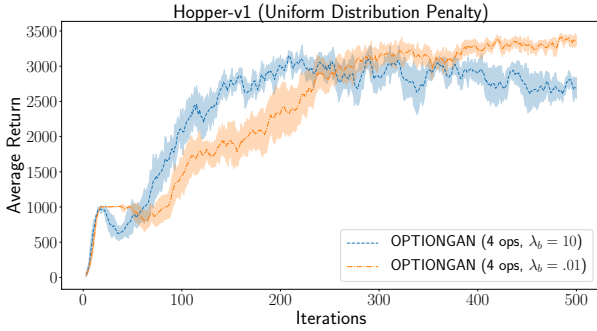
Figure 4: Effect of uniform distribution regularizer. Average π_Ω across final sample novice rollouts: $\lambda_b = 10.0$, $[.27, .21, .25, .25]$; $\lambda_b = .01$, $[0., 0., .62, .38]$.
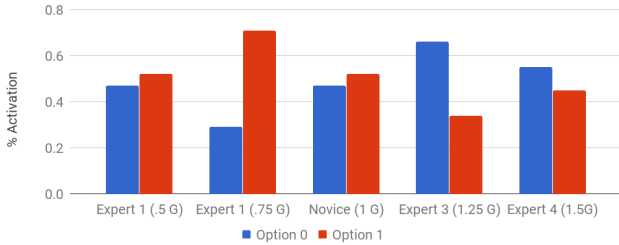

Figure 5: Probability distribution of π_Ω over options on expert demonstrations. Inherent structure is found in the underlying demonstrations. .75G demonstration state spaces are significantly assigned to Option 1 and similarly, 1.25G to Option 0.

the closer options (see supplemental material for supporting projected state space mappings). This indicates that the policy-over-options specializes over the experts and is thus inherently beneficial for use in one-shot transfer learning.

Discussion

We propose a direct extension of the options framework by adding joint reward-policy options. We learn these options in the context of generative adversarial inverse reinforcement learning and show that this method outperforms the single policy case in a variety of tasks – particularly in transfer settings. Furthermore, the learned options demonstrate temporal and interpretable cohesion without specifying a call-and-return termination function.

Our formulation of joint reward-policy options as a Mixture-Of-Experts allows for: potential upscaling to extremely large networks as in (Shazeer et al. 2017), reward shaping in forward RL, and using similarly specialized MoEs in generative adversarial networks.

This work presents an effective and extendable framework. Our optionated networks capture the problem structure effectively, which allows strong generalization in one-shot transfer learning. Moreover, as adversarial methods are now commonly used across a myriad of communities, we believe the embedding of options within this methodology is an excellent delivery mechanism to exploit the benefits of hierarchical RL in many new fields.

Acknowledgements

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, 1–. New York, NY, USA: ACM.

Bacon, P.-L.; Harb, J.; and Precup, D. 2016. The option-critic architecture. *arXiv preprint arXiv:1609.05140*.

Bengio, E.; Bacon, P.-L.; Pineau, J.; and Precup, D. 2015. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym.

Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; and Abbeel, P. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2172–2180.

Christiano, P.; Shah, Z.; Mordatch, I.; Schneider, J.; Blackwell, T.; Tobin, J.; Abbeel, P.; and Zaremba, W. 2016. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*.

Dietterich, T. G. 1998. The maxq method for hierarchical reinforcement learning. In *ICML*, 118–126.

Hausman, K.; Chebotar, Y.; Schaal, S.; Sukhatme, G.; and Lim, J. 2017. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. *arXiv preprint arXiv:1705.10479*.

Henderson, P.; Chang, W.-D.; Shkurti, F.; Hansen, J.; Meger, D.; and Dudek, G. 2017. Benchmark environments for multitask learning in continuous domains. *ICML Lifelong Learning: A Reinforcement Learning Approach Workshop*.

Ho, J., and Ermon, S. 2016. Generative adversarial imitation learning. *CoRR* abs/1606.03476.

Jacobs, R. A.; Jordan, M. I.; Nowlan, S. J.; and Hinton, G. E. 1991. Adaptive mixtures of local experts. *Neural Comput.* 3(1):79–87.

Kingma, D. P., and Ba, J. 2014. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*.

Krishnan, S.; Garg, A.; Liaw, R.; Miller, L.; Pokorny, F. T.; and Goldberg, K. 2016. Hirl: Hierarchical inverse reinforcement learning for long-horizon tasks with delayed rewards. *arXiv preprint arXiv:1604.06508*.

Liu, Y., and Yao, X. 2002. Learning and evolution by minimization of mutual information. In *International Conference on Parallel Problem Solving from Nature*, 495–504. Springer.

Merel, J.; Tassa, Y.; Srinivasan, S.; Lemmon, J.; Wang, Z.; Wayne, G.; and Heess, N. 2017. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 1928–1937.

Ng, A. Y., and Russell, S. J. 2000. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, 663–670. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Schulman, J.; Levine, S.; Moritz, P.; Jordan, M. I.; and Abbeel, P. 2015a. Trust region policy optimization. *CoRR* abs/1502.05477.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015b. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sermanet, P.; Xu, K.; and Levine, S. 2016. Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112(1-2):181–211.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, 5026–5033.

Wang, Z.; Merel, J.; Reed, S.; Wayne, G.; de Freitas, N.; and Heess, N. 2017. Robust imitation of diverse behaviors. *arXiv preprint arXiv:1707.02747*.

Ziebart, B. D.; Maas, A.; Bagnell, J. A.; and Dey, A. K. 2008. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, 1433–1438. AAAI Press.

Supplemental Material

Expanded Equations

The expectation over the discriminator loss for the option case can be expanded:

$$L_\Omega = \mathbb{E}_\omega \left[\pi_{\Omega,\zeta}(\omega, s) L_{\hat{\theta},\omega} \right] + L_{reg}$$
$$= \sum_\omega \pi_{\Omega,\zeta}(\omega, s) L_{\hat{\theta},\omega} + L_{reg} \tag{10}$$

As can the regularization terms:

$$L_b = \sum_\omega ||\mathbb{E}[\pi_\Omega(\omega)] - \tau||_2$$
$$\approx \sum_\omega ||\frac{1}{m_b} \sum_i^{m_b} (\pi_\Omega(\omega, s_i)) - \tau||_2 \tag{11}$$

$$L_e = \mathbb{E}\left[|| \left(\frac{1}{||\Omega||} \sum_\omega \pi_\Omega(\omega) \right) - \tau||_2 \right]$$
$$\approx \frac{1}{m_b} \sum_i^{m_b} || \left(\frac{1}{||\Omega||} \sum_\omega \pi_\Omega(\omega, s_i) \right) - \tau||_2 \tag{12}$$

$$L_v = - \sum_\omega var_\omega\{\pi_\Omega(s)\}$$
$$\approx - \sum_\omega \frac{1}{m_b} \sum_i^{m_b} \left(\pi_\Omega(\omega, s_i) - \left(\frac{1}{m_b} \sum_i^{m_b} \pi_\Omega(\omega, s_i) \right) \right) \tag{13}$$

Expert Collection

The expert demonstration rollouts (state sequences) for all OpenAI Gym (Brockman et al. 2016) environments were obtained from policies trained for 1000 iterations using Trust Region Policy Optimization (Schulman et al. 2015a) with parameters $KL_{max} = .01$, generalized advantage estimation $\lambda = .97$, discount factor $\gamma = .99$ and batch size $25,000$ (rollout timesteps shared when updating the discriminator and policy). For the Roboschool (Schulman et al. 2017) Flagrun-v1 environment, the rollouts were obtained using the PPO pre-trained expert provided with Roboschool.

Experimental Setup and Hyperparameters

Observations are **not** normalized in all cases as we found that it did not help or hurt performance. In all cases for advantage estimation we use a value approximator as in (Schulman et al. 2015a), which uses L-BFGS optimization with a mixing fraction of .1. That is, it uses the current prediction $V_\theta(s)$ and mixes it with the actual discounted returns with .1 belonging to the actual discounted returns and .9 belonging to the current prediction. This is identical to the original Trust Region Policy Optimization (TRPO) code as provided at: `https://github.com/joschu/modular_rl/`. We perform a maximum of 20 L-BFGS iterations per value function update. For all the environments, we let the agent act until the maximum allowed timesteps of the specific environment (as set by default in OpenAI Gym), gather the rollouts and keep the number of timesteps per batch desired.

For all policy optimization steps in both IRLGAN and OPTIONGAN, we use TRPO with the with parameters set to the same values as the ones used for the expert collection ($KL_{max} = .01$, generalized advantage estimation $\lambda = .97$, discount factor $\gamma = .99$ and batch size $25,000$), except for the Roboschool Flagrun Experiment where PPO was used instead, as explained in its respective section below.

Simple Tasks and Transfer Tasks

For simple tasks we use 10 expert rollouts while for transfer tasks we use 40 expert rollouts (10 from each environment variation).
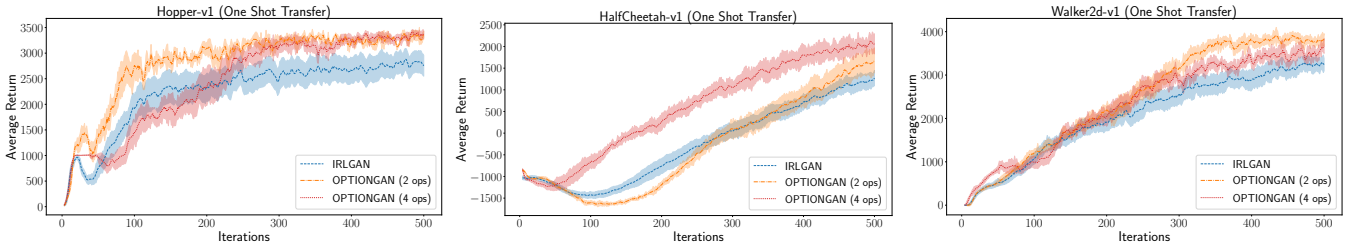
Figure 6: Experiments from 40 expert demonstrations from varied gravity environments without any demonstrations on the novice training environment. Average returns from expert demonstrations across all mixed environments: 3657.71 (Hopper-v1), 4181.97 (HalfCheetah-v1), 4218.12 (Walker2d-v1).
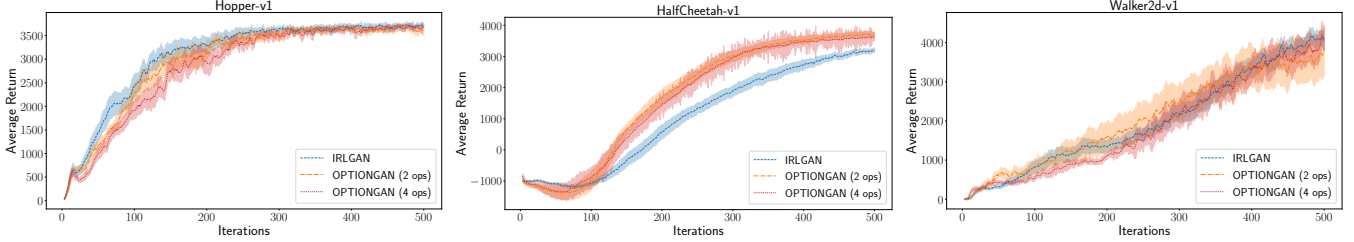

Figure 7: Simple locomotion task learning curves. The True Average Return provided by the expert demonstrations are: 3778.82 (Hopper-v1), 4156.94 (HalfCheetah-v1), 5528.51 (Walker2d-v1). Error bars indicate standard error of True Average Return across 10 trial runs.
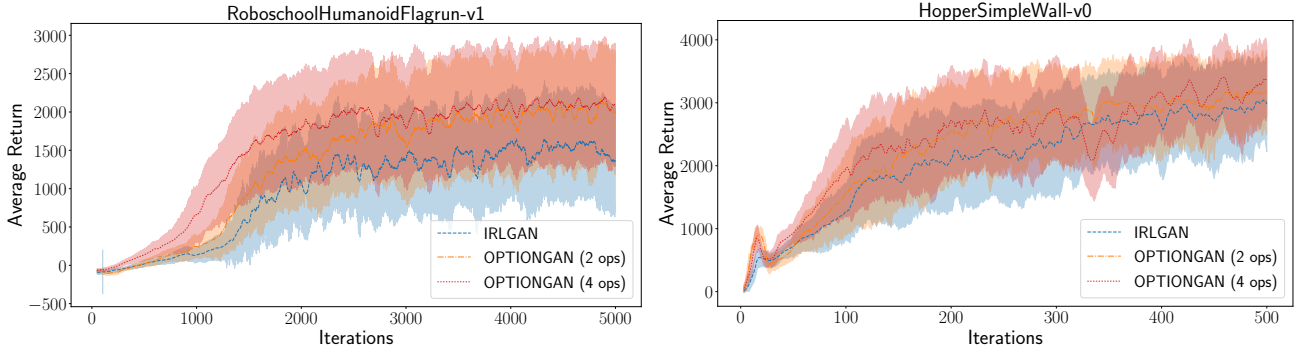.

Figure 8: Evaluation on RoboschoolHumanoidFlagrun-v1 environment (Schulman et al. 2017). Average expert rollout returns: 2822.13.

IRLGAN We use a Gaussian Multilayer Perceptron policy as in (Schulman et al. 2015a) with two 64 units hidden layers and tanh hidden layer activations. The output of the network gives the Gaussian mean and the standard deviation is modeled by a single learned variable as in (Schulman et al. 2015a). Similarly for our discriminator network, we use the same architecture with a sigmoid output, tanh hidden layer activations, and a learning rate of $1 \cdot 10^{-3}$ for the discriminator. We do not use entropy regularization or $l2$ regularization as it resulted in worse performance. For every policy update we perform 3 discriminator updates as we found the policy optimization step is able to handle this and results in faster learning.

OPTIONGAN Aligning with the IRLGAN networks, we make use of a Gaussian Multilayer Perceptron policy as in (Schulman et al. 2015a) with 2 hidden layers of 64 units with tanh hidden layer activations for our shared hidden layers. These hidden layers connect to $||\Omega||$ options depending on the experiment (2 or 4). In this case the output of the network gives the Gaussian mean for each option and the standard deviation is modeled by a single learned variable per option. The policy-over-options is also modeled by a 2 layered 64 units network with tanh activations and a softmax output corresponding to the number of options.

For our discriminator, we use the same architecture with tanh hidden layer activations, a sigmoid output and $||\Omega||$ outputs, one for each option. We use the policy over options to create a specialized mixture of experts model with a specialized loss

function which converges to options. We use a learning rate of $1 \cdot 10^{-3}$ for the discriminator.

Same as IRLGAN, we do not make use of entropy regularization or $l2$ regularization as we found either regularizers to hurt performance. Instead we use scaling factors for the regularization terms included in the loss: $\lambda_b = 10.0, \lambda_e = 10.0, \lambda_v = 1.0$ for the 2 options case and $\lambda_b = 0.01, \lambda_e = 10.0, \lambda_v = 1.0$ for the 4 options case. Again, we perform 3 discriminator updates per policy update

RoboschoolHumanoidFlagrun-v1

For Roboschool experiments we use proximal policy optimization (PPO) with a clipping objective (Schulman et al. 2017) (clipping parameter set to $\epsilon = .02$). We perform 5 Adam (Kingma and Ba 2014) policy updates on the PPO clipping objective with a learning rate of $1 \cdot 10^{-3}$. The value function and advantage estimation parameters from previous experiments are maintained while our network architecture sizes are increased to $(128, 128)$ and use ReLU activations instead of tanh.

Decomposition of Rewards over Expert Demonstrations

We show that the trained policy-over-options network shows some intrinsic structure over the expert demonstrations.
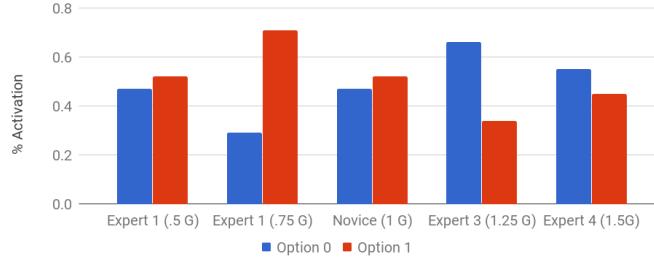

Figure 9: Probability distribution of π_Ω over options on expert demonstrations. Inherent structure is found in the underlying demonstrations.
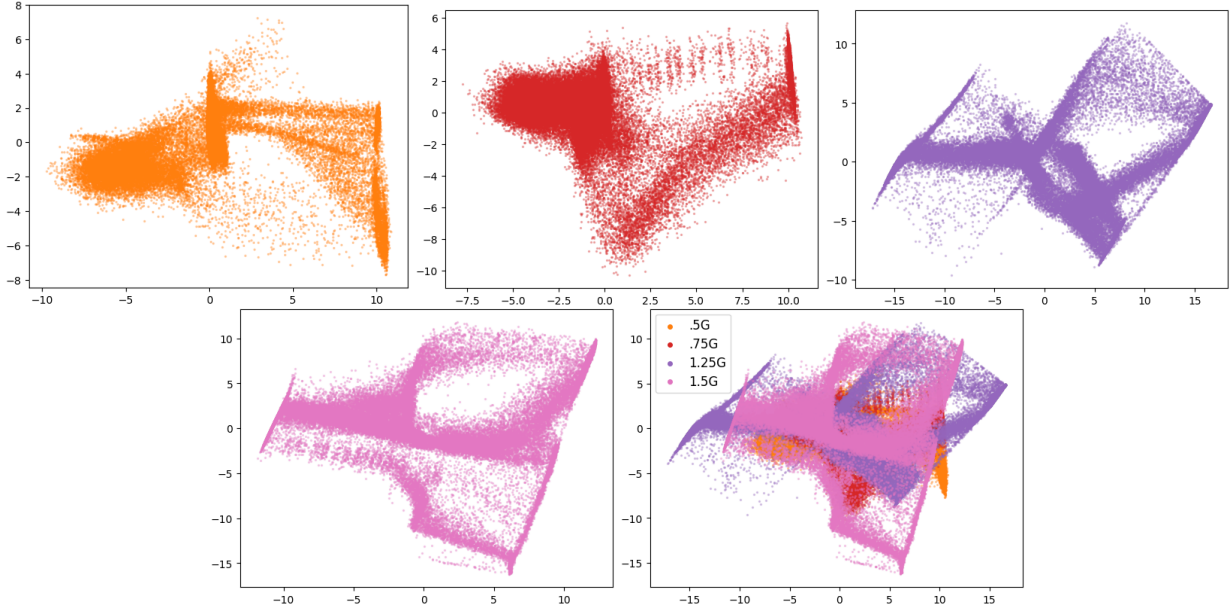

Figure 10: State distribution of expert demonstrations projected onto a 2D plane to pair with Figure 9. Axes correspond to projected state dimensions from the original 6-D state space using SVD.

In Figure 9 are shown the activation of the gating function across expert rollouts after training. We see that the underlying division in expert demonstrations is learned by the policy-over-options, which indicates that our method for training the policy-over-options induces it to learn a latent structure to the expert demonstrations and thus can benefit in the transfer case since each option inherently specializes to be used in different environments. We find that options specialized more clearly over the

experts with environments closest to the normal gravity environment, while the others use an even mixture of options. This is due to the fact that the mixing specialized options are able to cover the state space of the non-specialized options as we can observe from the state distribution of the expert demonstrations shown in Figure 10.