Directed-Info GAIL: Learning Hierarchical Policies from Unsegmented Demonstrations using Directed Information

Anonymous Author(s)
Affiliation
Address
`email`

Abstract: The use of imitation learning to learn a single policy for a complex task that has multiple modes or hierarchical structure can be challenging. In fact, previous work has shown that when the modes are known, learning separate policies for each mode or sub-task can greatly improve the performance of imitation learning. In this work, we discover the interaction between sub-tasks from their resulting state-action trajectory sequences using a directed graphical model. We propose a new algorithm based on the generative adversarial imitation learning framework which automatically learns sub-task policies from unsegmented demonstrations. Our approach maximizes the directed information flow in the graphical model between sub-task latent variables and their generated trajectories. We also show how our approach connects with existing 'Options' framework commonly used to learn hierarchical policies.

1 Introduction

Complex human activities can often be broken down into various simpler sub-activities or sub-tasks that can serve as the basic building blocks for completing a variety of complicated tasks. For instance, when driving a car, a driver may perform several simpler sub-tasks such as driving straight in a lane, changing lanes, executing a turn and braking, in different orders and for varying times depending on the source, destination, traffic conditions etc. Using imitation learning to learn one monolithic policy to represent an activity can be challenging as it does not make explicit the sub-structure between the parts of the activity. In this work, we develop an imitation learning framework that can learn a policy for each of these sub-tasks given unsegmented activity demonstrations and also learn a macro-policy which dictates switching from one sub-task policy to another. Learning sub-task specific policies has the benefit of shared learning. Each such sub-task policy also needs to specialize over a restricted state space, thus making the learning problem easier.

Previous works in imitation learning [1, 2] focus on learning each sub-task specific policy using *segmented* expert demonstrations by modeling the variability in each sub-task policy using a latent variable. This latent variable is inferred by enforcing high mutual information between the latent variable and expert demonstrations. This information theoretic perspective is equivalent to the graphical model shown in Figure 1 (Left), where the node c represents the latent variable. However, since learning sub-task policies requires isolated demonstrations for each sub-task this setup is difficult to scale to many real world scenarios where providing such segmented trajectories is cumbersome. Further, this setup does not learn a macro-policy to combine the learned sub-task policies in meaningful ways to achieve different tasks.

In our work, we aim to learn each sub-task policy directly from *unsegmented* activity demonstrations. For example, given a task consisting of three sub-tasks — A, B and C, we wish to learn a policy to complete sub-task A, learn when to transition from A to B, finish sub-task B and so on. To achieve this we use a causal graphical model, which can be represented as a Dynamic Bayesian Network as shown in Figure 1 (Right). The nodes c_t denote latent variables which indicate the currently active sub-task and the nodes τ_t denote the state-action pair at time t. We consider as given, a set of expert demonstrations, each of which is represented by $\boldsymbol{\tau} = \{\tau_1, \cdots, \tau_T\}$ and has a corresponding

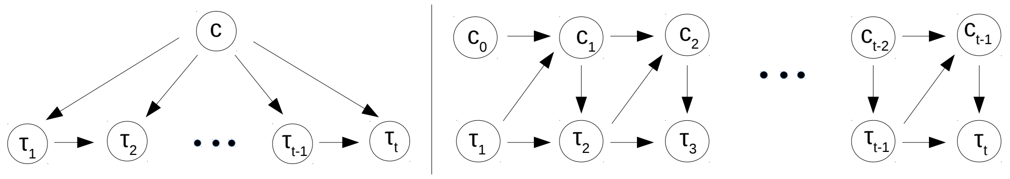


Figure 1: **Left:** Graphical model used in Info-GAIL [1]. **Right:** Causal model in this work. The latent code causes the policy to produce a trajectory. The current trajectory, and latent code produce the next latent code

sequence of latent factors $\boldsymbol{c} = \{c_1, \cdots, c_{T-1}\}$. The sub-activity at time t dictates what state-action pair was generated at time t. The previous sub-task and the current state-action pair together cause the selection of the next sub-task.

As we will discuss in Section 3, extending the use of mutual information to learn sub-task policies from unsegmented demonstrations is *problematic*, as it requires learning the macro-policy as a conditional probability distribution which depends on the *unobserved future*. This unobserved future is unknown during earlier points of interaction (Figure 1). To alleviate this, in our work we aim to force the policy to generate trajectories that maximize the directed information or causal information [3] flow from trajectories to latent factors of variation within the trajectories instead of mutual information. Using directed information requires us to learn a causally conditioned probability distribution [4] which depends only on the *observed past* while allowing the unobserved future to be sequentially revealed. Further, since there exists *feedback* in our causal graphical model *i.e.*, information flows from the latent variables to trajectories and vice versa, directed information also provides a better upper bound on this information flow between the latent variables and expert trajectories than does the conventional mutual information [3, 4].

We also draw connections with existing work on learning sub-task policies using imitation learning with the *options framework* [5, 6]. We show that our work, while derived using the information theoretic perspective of maximizing directed information, bears a close resemblance to applying the options framework in a generative adversarial imitation setting. Thus, our approach combines the benefits of learning hierarchical policies using the options framework with the robustness of generative adversarial imitation learning, helping overcome problems such as compounding errors that plague behaviour cloning.

In summary, the main contributions of our work include:

- We extend existing generative adversarial imitation learning frameworks to allow for learning of sub-task specific policies by maximizing directed information in a causal graph of sub-activity latent variables and observed trajectory variables.
- We draw connections between previous works on imitation learning with sub-task policies using *options* and show that our proposed approach resembles learning of *options* in a generative adversarial setting.
- We show through experiments on both discrete and continuous state-action spaces, the ability of our approach to segment expert demonstrations into meaningful sub-tasks and combine sub-task specific policies to perform the desired task.

2 Related Work

2.1 Imitation Learning

Imitation Learning [7] aims at learning policies that can mimic expert behaviours from demonstrations. Modeling the problem as a Markov Decision Process (MDP), the goal in imitation learning is to learn a policy $\pi(a|s)$, which defines the conditional distribution over actions $a \in \mathcal{A}$ given the state $s \in \mathcal{S}$, from state-action trajectories $\tau = (s_0, a_0, \cdots, s_T)$ of expert behaviour. Recently, [8] introduced an imitation learning framework called Generative Adversarial Imitation Learning (GAIL) that is able to learn policies for complex high-dimensional physics-based control tasks. They reduce the imitation learning problem into an adversarial learning framework, for which they utilize Generative

Adversarial Networks (GAN) [9]. The generator network of the GAN represents the agent's policy π while the discriminator network serves as a local reward function and learns to differentiate between state-action pairs from the expert policy $\pi_{\mathbb{E}}$ and from the agent's policy π. Mathematically, it is equivalent to solving the following optimization problem,

$$\min_{\pi} \max_{D} \mathbb{E}_{\pi}[\log D(s,a)] + \mathbb{E}_{\pi_E}[1 - \log D(s,a)] - \lambda H(\pi)$$

InfoGAIL [1] and [2] solve the problem of learning from policies generated by a mixture of experts. They introduce a latent variable c into the policy function $\pi(a|s,c)$ to separate different type of behaviours present in the demonstration. To incentivize the network to use the latent variable, they utilize an information-theoretic regularization enforcing that there should be high mutual information between c and the state-action pairs in the generated trajectory, a concept that was first introduced in InfoGAN [10]. They introduce a variational lower bound $L_1(\pi, Q)$ of the mutual information $I(c;\tau)$ to the loss function in GAIL.

$$L_1(\pi, Q) = \mathbb{E}_{c\sim p(c), a\sim\pi(\cdot|s,c)} \log Q(c|\tau) + H(c) \leq I(c;\tau)$$

The modified objective can then be given as,

$$\begin{aligned} \min_{\pi,q} \max_{D} \mathbb{E}_{\pi}[\log D(s,a)] + \mathbb{E}_{\pi_E}[1 - \log D(s,a)] \\ -\lambda_1 L_1(\pi, q) - \lambda_2 H(\pi) \end{aligned}$$

InfoGAIL models variations between different trajectories as the latent codes correspond to trajectories coming from different demonstrators. In contrast, we aim to model *intra-trajectory variations* and latent codes in our work correspond to sub-tasks (variations) within a demonstration. In section 3, we discuss why using a mutual information based loss is in-feasible in our problem setting and describe our proposed approach.

2.2 Options

Consider an MDP with states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$. Under the options framework [5], an option, indexed by $o \in \mathcal{O}$ consists of a sub-policy $\pi(a|s,o)$, a termination policy $\pi(b|s,\bar{o})$ and an option activation policy $\pi(o|s)$. After an option is initiated, actions are generated by the sub-policy until the option is terminated and a new option is selected. In [6] the authors formulate the options framework as a probabilistic graphical model where options are treated as latent variables which are then learned from expert data. The option policies ($\pi(a|s,o)$) are analogous to sub-task policies in our work, we connect our work to this existing method and show that our method can be seen as a generative adversarial variant of the approach in [6].

3 Proposed Approach

As mentioned in the previous section, while prior approaches can learn to disambiguate the multiple modalities in the demonstration of a sub-task and learn to imitate them, they cannot learn to imitate demonstrations of unsegmented long tasks that are formed by a combination of many small sub-tasks. To learn such sub-task policies from unsegmented gestures we use the graphical model in Figure 1 (Right), *i.e.*, consider a set of expert demonstrations, each of which is represented by $\boldsymbol{\tau} = \{\tau_1, \cdots, \tau_T\}$ where τ_t is the state-action pair observed at time t. Each such demonstration has a corresponding sequence of latent variables $\boldsymbol{c} = \{c_1, \cdots, c_{T-1}\}$ which denote the sub-activity in the demonstration at any given time step.

As noted before, previous approaches [1, 2] model the expert sub-task demonstrations using only a single latent variable. To enforce the model to use this latent variable, previous approaches propose to maximize the mutual information between the demonstrated sequence of state-action pairs and the latent embedding of the nature of the sub-activity. This is achieved by adding a lower bound to the mutual information between the latent variables and expert demonstrations. This variational lower bound of the mutual information is then combined with the the adversarial loss for imitation learning proposed in [8]. Extending this to our setting, where we have a *sequence* of latent variables $\boldsymbol{c}$, yields the following lower bound on the mutual information,

$$\begin{aligned} L(\pi, q) = \sum_t \mathbb{E}_{c^{1:t}\sim p(c^{1:t}), a^{t-1}\sim\pi(\cdot|s^{t-1}, c^{1:t-1})} \Big[\\ \log q(c^t | c^{1:t-1}, \boldsymbol{\tau}) \Big] + H(\boldsymbol{c}) \le I(\boldsymbol{\tau}; \boldsymbol{c}) \end{aligned} \tag{1}$$

Observe that the dependence of q on the entire trajectory $\boldsymbol{\tau}$ precludes the use of such a distribution at test time, where only the trajectory up to the current time is known. To overcome this limitation, in this work we propose to force the policy to generate trajectories that maximize the *directed* or *causal* information flow from trajectories to the sequence of latent sub-activity variables instead. As we show below, by using directed information instead of mutual information, we can replace the dependence on $\boldsymbol{\tau}$ with a dependence on the trajectory generated up to current time t.

The directed information flow from a sequence $\boldsymbol{X}$ to $\boldsymbol{Y}$ is given by,

$$I(\boldsymbol{X} \to \boldsymbol{Y}) = H(\boldsymbol{Y}) - H(\boldsymbol{Y} \| \boldsymbol{X})$$

where $H(\boldsymbol{Y}\|\boldsymbol{X})$ is the causally-conditioned entropy. Replacing $\boldsymbol{X}$ and $\boldsymbol{Y}$ with the sequences $\boldsymbol{\tau}$ and $\boldsymbol{c}$ gives,

$$\begin{aligned} I(\boldsymbol{\tau} \to \boldsymbol{c}) &= H(\boldsymbol{c}) - H(\boldsymbol{c}\|\boldsymbol{\tau}) \\ &= H(\boldsymbol{c}) - \sum_t H(c^t | c^{1:t-1}, \tau^{1:t}) \\ &= H(\boldsymbol{c}) + \sum_t \sum_{c^{1:t-1}, \tau^{1:t}} \Big[p(c^{1:t-1}, \tau^{1:t}) \\ &\sum_{c^t} p(c^t | c^{1:t-1}, \tau^{1:t}) \log p(c^t | c^{1:t-1}, \tau^{1:t}) \Big] \end{aligned} \tag{2}$$

Here $\tau^{1:t} = (s_1, \cdots, a_{t-1}, s_t)$. A variational lower bound, $L_1(\pi, q)$ of the directed information, $I(\boldsymbol{\tau} \to \boldsymbol{c})$ which uses an approximate posterior $q(c^t|c^{1:t-1}, \tau^{1:t})$ instead of the true posterior $p(c^t|c^{1:t-1}, \tau^{1:t})$ can then be derived to get,

$$\begin{aligned} L_1(\pi, q) = \sum_t \mathbb{E}_{c^{1:t}\sim p(c^{1:t}), a^{t-1}\sim\pi(\cdot|s^{t-1}, c^{1:t-1})} \Big[\\ \log q(c^t | c^{1:t-1}, \tau^{1:t}) \Big] + H(\boldsymbol{c}) \le I(\boldsymbol{\tau} \to \boldsymbol{c}) \end{aligned} \tag{3}$$

Thus, by maximizing directed information instead of mutual information, we can learn a posterior distribution over the next latent factor c given the latent factors discovered up to now and the trajectory followed up to now, thereby removing the dependence on the future trajectory. In practice, we do not consider the $H(\boldsymbol{c})$ term. This gives us the objective,

$$\begin{aligned} \min_{\pi, q} \max_{D} \ \mathbb{E}_{\pi}[\log D(s, a)] + \mathbb{E}_{\pi_E}[1 - \log D(s, a)] \\ - \lambda_1 L_1(\pi, q) - \lambda_2 H(\pi) \end{aligned} \tag{4}$$

We call this approach Directed-Info GAIL. Notice, that to compute the loss in equation (3), we need to sample from the prior distribution $p(c^{1:t})$. In order to estimate this distribution, we first pre-train a variational auto-encoder (VAE) [11] on the expert trajectories. We describe the details of the VAE step in the next sub-section.

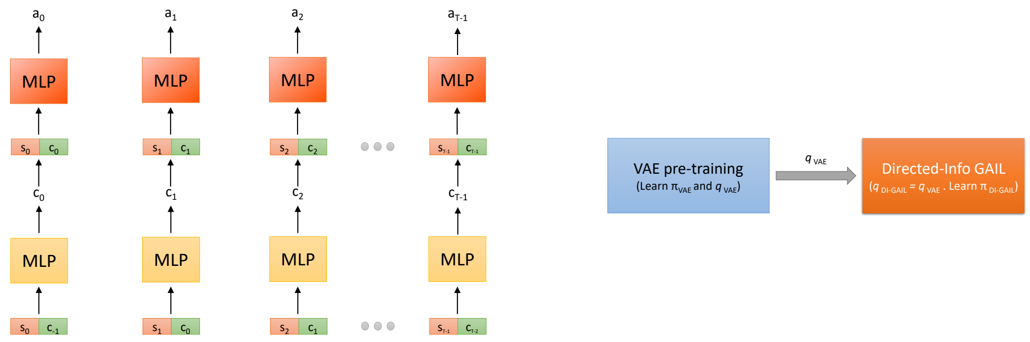


Figure 2: Left: VAE pre-training step. The VAE encoder uses the current state (s_t), and previous latent variable (c_{t-1}) to produce the current latent variable (c_t). The decoder reconstructs the action (a_t) using s_t and c_t. Right: An overview of the proposed approach. We use the VAE pre-training step to learn an approximate prior over the latent variables and use this to learn sub-task policies in the proposed Directed-Info GAIL step.

3.1 VAE pre-training

Figure 2 (left) shows the design of the VAE pictorially. The VAE consists of two multi-layer perceptrons that serve as the encoder and the decoder. The encoder uses the current state s_t and the previous latent variable c_{t-1} to produce the current latent variable c_t. We used the Gumbel-softmax trick [12] to obtain samples of latent variables from a categorical distribution. The decoder then takes s_t and c_t as input and produces the action a_t as output. We use the following objective, which maximizes the lower bound of the probability of the trajectories $p(\boldsymbol{\tau})$, to train our VAE,

$$\begin{aligned} L_{\text{VAE}}(\pi, q; \boldsymbol{\tau_i}) = &-\sum_t \mathbb{E}_{c^t \sim q}\Big[\log \pi(a^t|s^t, c^{1:t})\Big] \\ &+ \sum_t D_{\text{KL}}(q(c^t|c^{1:t-1}, \tau^{1:t}) \| p(c^t|c^{1:t-1})) \end{aligned} \tag{5}$$

Figure 2 (right) gives an overview of the complete method. The VAE pre-training step allows us to get approximate samples from the distribution $p(c^{1:t})$ to optimize (4). This is done by using q to obtain samples of latent variable sequence $\boldsymbol{c}$ by using its output on the expert demonstrations. In practice, we fix the weights of the network q to those obtained from the VAE pre-training step when optimizing the Directed-Info GAIL loss in equation (4).

3.2 Connection with options framework

In [6] the authors provide a probabilistic perspective of the options framework. Although, [6] consider separate termination and option latent variables (b^t and o^t), for the purpose of comparison, we collapse them into a single latent variable c^t, similar to our framework with a distribution $p(c^t|s^t, c^{t-1})$. The lower-bound derived in [6] which is maximized using Expectation-Maximization (EM) [13] can then be written as (suppressing dependence on parameters),

$$\begin{aligned} p(\tau) \geq &\sum_t \sum_{c^{t-1:t}} p(c^{t-1:t}|\tau) \log p(c^t|s^t, c^{t-1})) \\ &+ \sum_t \sum_{c^t} p(c^t|\tau) \log \pi(a^t|s^t, c^t) \end{aligned} \tag{6}$$

Note that the first term in Equation (6) *i.e.*, the expectation over the distribution $\log p(c^t|s^t, c^{t-1})$ is the same as Equation (3) of our proposed approach with a one-step Markov assumption and a conditional expectation with given expert trajectories instead of an expectation with generated trajectories. The second term in Equation (6) *i.e.*, the expectation over $\log \pi(a^t|s^t, c^t)$ is replaced by the GAIL loss in Equation (4). Our proposed Directed-Info GAIL can be therefore be considered as the generative adversarial variant of imitation learning using the options framework. The VAE behaviour cloning pre-training step in Equation (5) is exactly equivalent to Equation (6), where we use approximate variational inference using VAEs instead of EM. Thus, our approach combines the benefits of both behavior cloning and generative adversarial imitation learning. Using GAIL enables learning of robust policies that do not suffer from the problem of compounding errors. At the same time, conditioning GAIL on latent codes learned from the behavior cloning step prevents the issue of mode collapse in GANs.

4 Experiments

We present results on both discrete and continuous state-action environments. In both of these settings we show that our method is able to segment out sub-tasks from given expert state-action trajectories, learn sub-task conditioned policies, and learn to combine these sub-tasks in order to achieve the task objective. For comparative analysis we compare our results with both generative adversarial imitation learning approach [8] and the supervised behavior cloning approaching using a VAE.

4.1 Discrete Environment

As our discrete environment we choose a grid world environment which consists of a 15×11 grid and four rooms connected via corridors as shown in Figures 3 and 4. The agent spawns at a random location in the grid at the beginning of each episode. One of the four rooms is then selected at random and an apple is placed at the centre of the room. The goal of the agent is to find the shortest path to the apple using four actions — up, down, left and right. Expert trajectories were created using Dijkstra's shortest path algorithm.

4.2 Continuous Environments

We experiment with two continuous state-action environments. The goal in the first task is to swing-up an underpowered pendulum. The agent must learn to exert appropriate torques, swinging-up the pendulum and keeping it in an upright position. We used the Pendulum-v0 environment defined in OpenAI Gym [14]. The states $s \in \mathcal{R}^2$ and the actions $a \in \mathcal{R}$.

In the second task, the agent must learn to balance an inverted pendulum on a cart with the aim of keeping it in a stable position and prevent it from falling. We used OpenAI Gym's InvertedPendulum-v2 environment which uses the MuJoCo physics engine [15] for this task. The states $s \in \mathcal{R}^4$ and the actions $a \in \mathcal{R}$.

To generate expert trajectories we train an RL agent using Proximal Policy Optimization [16]. We used 25 expert trajectories for all of our experiments.

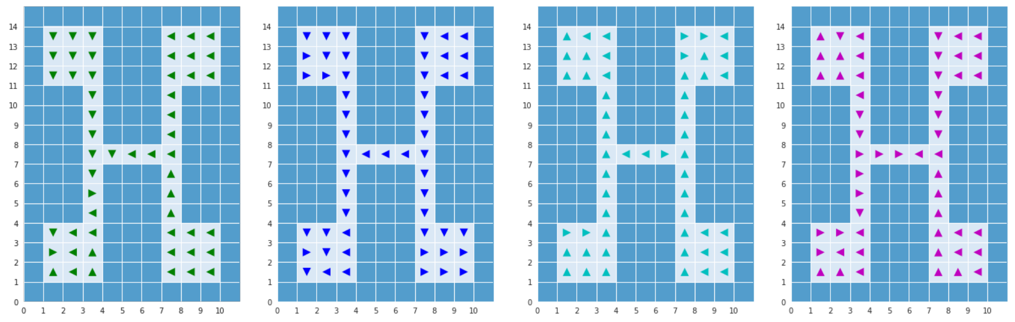
4.3 Implementation Details

We use multi-layer perceptrons for our policy (Generator), value, and reward (Discriminator) function representations. Each network consisted of 2 hidden layers with 64 units in each layer and 'tanh' as our non-linearity function. We used the Adam [17] as our optimizer setting an initial learning rate of $3e^{-4}$. Further, we used the Proximal Policy Optimization algorithm [16] to train our policy network with $\epsilon = 0.2$

4.4 Results

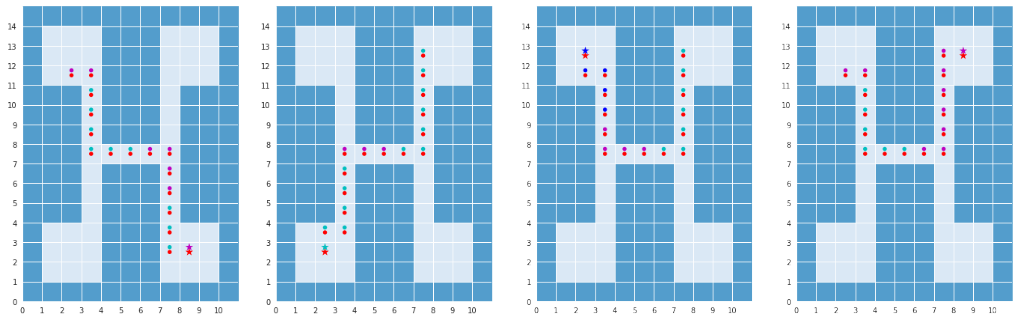
Figure 3 shows sub-task policies learned by our approach in the discrete task. Each of the four plots corresponds to a different value of the latent variable. The arrow at every state in the grid in each plot shows the agent action (direction) with the highest probability in that state for that latent variable. In the discussion that follows, we label the rows from 1 to 4 starting from the room at the top left and moving in the clockwise direction.

Notice how the different latent variables correspond to different sub-tasks. For e.g., the latent code in Figure 3(b) learns the sub-task of moving from room 1 to room 3 and from room 2 to room 4 and



(a) (b) (c) (d)

Figure 3: Results on the Four Rooms environment. Each figure shows results for a different latent variable. The arrows in each cell indicate the direction (action) with highest probability in that state and using the given latent variable.



(a) (b) (c) (d)

Figure 4: Expert and generated trajectories in the Four Rooms environment. Star (*) represents the start state. The expert trajectory is shown in red. The color of the generated trajectory represents the latent code used by the policy at each time step.

the code in Figure 3(c) learns the opposite sub-tasks. The code in Figure 3(d) learns the sub-task of moving from rooms 2 and 4 to the horizontal corridor.

Figure 4 shows some examples of how the macro-policy switches between various latent codes to achieve the desired goals of reaching the apples in rooms 1 and 2 respectively.

Figure 5 shows sub-task policies learned by our approach on the Pendulum-v0 environment. The figure on the left shows the torque inputs applied by an agent trained using the proposed Directed-Info GAIL approach for different states (position and velocity). For the same set of states, the figure on the right shows the different sub-activity latent variables predicted by our model.

As can be seen in the above figure the network is able to correspond different latent variables for different sub-tasks. For instance, states that require an application of high torque are assigned a particular latent variable (shown in blue). Similarly, states that lie close to position 0 and require low

Environment	GAIL [8]	VAE	Directed-Info GAIL
Pendulum-v0	-121.42 ± 94.13	-142.89 ± 95.57	-125.39 ± 103.75
InvertedPendulum-v2	1000.0 ± 15.23	218.8 ± 7.95	1000.0 ± 14.97

Table 1: A comparison of returns for continuous environments. The returns were computed using 300 episodes. Our approach gives comparable returns to using GAIL but also segments expert demonstrations into sub-tasks. The proposed Directed-Info GAIL approach improves over the policy learned from the VAE pre-training step.

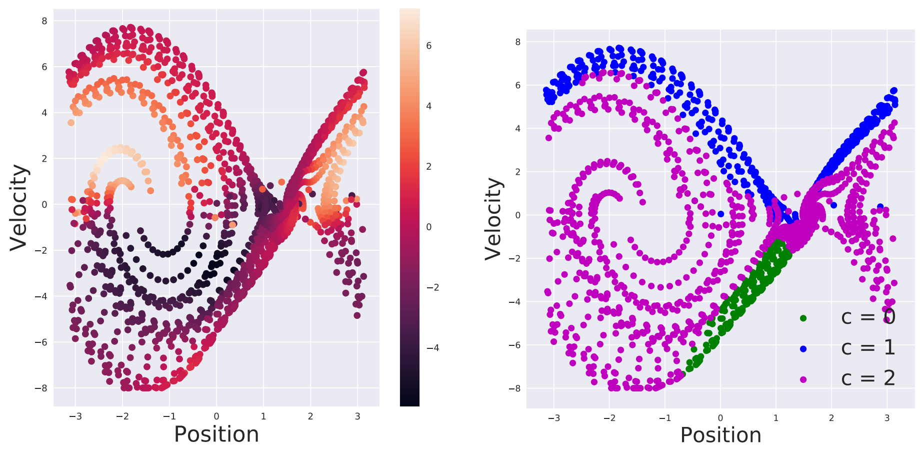


Figure 5: Results for Directed-Info GAIL on Pendulum environment. *Left:* Torque (action) output for different states by an agent trained using our approach. *Right:* Sub-activity latent variables as inferred by Directed-Info GAIL. Different colors represent different context.

torque (i.e. the desired target position) get assigned another latent variable (shown in green). The remaining states get classified as a separate sub-task.

Table 1 shows the mean and standard deviations of the returns over 300 episodes on the Pendulum-v0 and InvertedPendulum-v2 tasks. Our approach improves the performance over the VAE pre-training step, overcoming the issue of compounding errors. The performance is comparable to training with the vanilla GAIL loss but has the advantage of also segmenting the expert demonstrations into sub-tasks.

5 Conclusion

Learning separate sub-task policies can help improve the performance of imitation learning when the demonstrated task is complex and has a hierarchical structure. In this work, we present an algorithm that infers these latent sub-task policies directly from given unstructured and unlabelled expert demonstrations. We model the problem of imitation learning as a directed graph with sub-task latent variables and observed trajectory variables. We show that extending previous works on maximizing mutual information between the latent and observed variables is problematic as leads to a dependence on future unobserved variables when inferring the next latent variable. Our approach overcomes this by using the notion of directed information. We propose a generative adversarial imitation learning framework that learns sub-task and macro policies by maximizes the directed information flow between sub-task latent variables and given expert demonstrations. We further show theoretical connections with the options literature as used in hierarchical reinforcement and imitation learning. We evaluated our method on both discrete and continuous environments. Our experiments show that our method is able to segment the expert demonstrations into different sub-tasks, learn sub-task specific policies and then learn a macro-policy that can combines these sub-task. We further compare our method to two different baseline algorithms and show that our method is able to achieve comparable or better performance in all environments.

References

[1] Y. Li, J. Song, and S. Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In *Advances in Neural Information Processing Systems*, pages 3815–3825, 2017.

[2] K. Hausman, Y. Chebotar, S. Schaal, G. Sukhatme, and J. J. Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 1235–1245, 2017.

[3] J. Massey. Causality, feedback and directed information. In *Proc. Int. Symp. Inf. Theory Applic.(ISITA-90)*, pages 303–305. Citeseer, 1990.

[4] G. Kramer. *Directed information for channels with feedback*. PhD thesis, Eidgenossiche Technische Hochschule Zurich, 1998.

[5] R. S. Sutton, D. Precup, and S. P. Singh. Intra-option learning about temporally abstract actions. In *ICML*, volume 98, pages 556–564, 1998.

[6] C. Daniel, H. Van Hoof, J. Peters, and G. Neumann. Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104(2-3):337–357, 2016.

[7] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[8] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

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

[11] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[12] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[13] T. K. Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13 (6):47–60, 1996.

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

[15] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

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

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.