Abstract:
Imitation learning is the problem of teaching an agent to replicate expert policy from demonstrations when the underlying reward function is unavailable. This task becomes particularly challenging when the expert demonstrates a mixture of behaviors, often modeled by a discrete or continuous latent variable. Prior work has addressed imitation learning in such mixture scenarios by recovering the underlying latent variables, in the context of both supervised learning (behavior cloning), and generative adversarial imitation learning (GAIL). In several robotic locomotion tasks, simulated in the MuJoCo platform, we observe that existing models fail in distinguishing and imitating different modes of behavior in both cases of discrete and continuous latent variables. To address this problem, we introduce a novel generative model for behavior cloning, in a mode-separating manner. We also integrate our model with GAIL, to achieve robustness to the problem of compounding error caused by unseen states. We show that our models outperform the state-of-the-art in aforementioned experiments.

Keywords: Imitation Learning, Multimodality, Behavior Cloning, Adversarial Training

1 Introduction
The goal of imitation learning is to learn to perform a task from expert trajectories in the absence of a reward signal. There are mainly two viable approaches for this setting. The first approach, known as behavior cloning (BC) [1], learns the mapping between the expert states-action pairs in a supervised manner. While in this approach the mapping between individual state-action pairs can be accurately learned, the long-range dynamics within a trajectory are neglected, leading to problem of compounding error [2, 3]. Specifically, when the agent slightly diverges from the states visited by the expert in the demonstrations, it could take erroneous actions, and the error quickly accumulates as the policy unrolls. Therefore, BC only succeeds with large amounts of data. An alternative approach, known as inverse reinforcement learning (IRL), recovers a reward function under which the expert policy is optimal. Most algorithms for IRL, however, are iterative, and require calculating an optimal policy in each iteration, making them computationally expensive [4]. Generative adversarial imitation learning (GAIL) [5] is an algorithm that bypasses any intermediate reinforcement learning step. The goal of GAIL is to directly optimize for a policy trained by running reinforcement learning on a cost function that is jointly learned by maximum causal entropy IRL [4, 6]. The resulting algorithm is closely connected to generative adversarial networks [7], and can fit distributions of states and actions induced by expert behavior. In contrast to BC, GAIL has shown promise in sample efficiency in the number of expert trajectories, and in robustness to unseen states. In many imitation learning scenarios, expert demonstrations of the same task may exhibit different varieties while starting from the same initial state. This might be the result of multiple human experts performing the same task in different ways, or the existence of multiple unlabeled objectives, e.g. a robotic arm manipulating an object in different ways. In these mixture scenarios, both BC and GAIL lack an explicit scheme to distinguish the underlying individual modes. In behavior cloning, this may lead to an averaging of the actions across the modes. Moreover, adversarial training suffers from mode collapse, and the learned policy in GAIL often captures only a subset of control behaviors [8, 9].
Several prior works tried to address imitation of diverse behaviors. In [10, 11], expert skill labels are used in training, which differ from this work in that we assume expert modes are unlabeled. In InfoGAIL [8] and Intention-GAN [12], GAIL objective is augmented with mutual information between generated trajectories and their corresponding latent code via an auxiliary posterior network. In [9], a Variational Autoencoder (VAE) model is first used to encode expert trajectories using a low-dimensional continuous latent variable. The policy and the discriminator of GAIL are then both conditioned on the latent codes to perform imitation learning. However, as we benchmark in Section 4, these models can perform poorly in several scenarios.

In this work, we present a method for multimodal behavior cloning which spares us the difficulty of encoding trajectories. In particular, we introduce a generative model that fits distributions without any VAE-like encoder module. Related “encoder-free” frameworks have been recently considered for image synthesis [13, 14]. We further combine this model—as a behavior cloning method—with GAIL, to get the best of both worlds. This attempt is inspired by empirical study [15], in which it is demonstrated that combination of BC with GAIL (in a unimodal setting) achieves better performance in fewer training iterations.

In a nutshell, the main focus of this work is diverse imitation learning with simultaneous mode separation and accurate replication of expert behavior. Our contributions are twofold:

1. Introducing a novel generative model, named Self-Organizing Generative Model (SOG), for multimodal imitation learning. We provide a theoretical analysis demonstrating that SOG optimizes the maximum likelihood objective.
2. Integrating behavior cloning into the GAIL objective function to benefit from accuracy of SOG and robustness of GAIL at the same time. Thus, the resulting model (a) can generate trajectories that are faithful to the expert, (b) is robust to unseen states, (c) learns a latent variable over variations of expert policy without requiring to encode trajectories, and (d) captures all modes successfully without mode collapse.

2 Background

2.1 Preliminaries

We consider the conventional notion of an infinite-horizon discounted Markov decision process (MDP), and represent it with a tuple \((S, A, P, r, \rho_0, \gamma)\). Namely, \(S\) denotes the state space, \(A\) denotes the action space, \(P : S \times A \times S \rightarrow \mathbb{R}\) represents the state transition probability distribution, \(r : S \times A \rightarrow \mathbb{R}\) is the reward function, \(\rho_0 : S \rightarrow \mathbb{R}\) represents the distribution over initial states, and \(\gamma \in (0, 1)\) is the reward discount factor. Let \(\pi : S \times A \rightarrow \mathbb{R}\) denote a stochastic policy function.

Demonstrations of the policy \(\pi\) are sequences of state-action pairs, sampled as follows: \(s_0 \sim \rho_0, a_1 \sim \pi(a|s_1), s_{t+1} \sim P(s_{t+1}|s_t, a_t)\). With states and actions generated by policy \(\pi\), we define the expected return as

\[
E_\pi [r(s, a)] := \mathbb{E}_\pi [\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)].
\]

2.2 Imitation Learning

The goal of imitation learning is to replicate expert policy from demonstrations without access to the underlying reward function. There exists two approaches to this problem: (a) behavior cloning [1], which treats state-action pairs as inputs and outputs of a policy function, and optimizes for the \(L_2\) loss between expert actions and predicted actions; and (b) generative adversarial imitation learning [5] which optimizes the following objective:

\[
\min_{\pi_\theta} \max_D \mathbb{E}_{\pi_\theta} [\log D(s, a)] + \mathbb{E}_{\pi_E} [\log (1 - D(s, a))] - \lambda H(\pi_\theta)
\]

where \(\pi_\theta\) is the learned policy, implemented by a neural network with weights \(\theta\), \(\pi_E\) is the imitated policy, \(D\) is a discriminative classifier that distinguishes between state-action pairs from the trajectories generated by \(\pi_\theta\) and \(\pi_E\), and \(H(\pi_\theta) := \mathbb{E}_{\pi_\theta} [-\log \pi_\theta(a|s)]\) is the discounted causal entropy of the policy \(\pi_\theta\) [16]. GAIL bypasses learning of a reward function using the discriminator to guide \(\pi_\theta\) towards imitating \(\pi_E\). Optimization of GAIL involves alternating between a gradient ascent step to increase (1) with respect to the discriminator parameters, and a Trust Region Policy Optimization (TRPO) [17] or Proximal Policy Optimization (PPO) [18] step to decrease it with respect to the policy parameters.

TRPO and PPO take the biggest possible improvement step on a policy without stepping so far that performance accidentally deteriorates. In PPO, this is achieved by optimizing the following
"surrogate" objective at $i^{th}$ iteration:

$$\mathcal{L}_{PPO}(s, a, \theta_i, \theta) = \min \left( \frac{\pi_{\theta_i}(a|s)}{\pi_{\theta}(a|s)} A^{\pi_i}(s, a), \ g(e, A^{\pi_i}(s, a)) \right), \tag{2}$$

where $A$ is the advantage function [19], and $g$ is a "clipping" function for some small parameter $\epsilon > 0$:

$$g(e, A) = \begin{cases} (1 + \epsilon)A & A \geq 0 \\ (1 - \epsilon)A & A < 0. \end{cases} \tag{3}$$

2.3 Diverse Expert Trajectories

We now turn our attention to the situation where expert trajectories have different varieties. We assume that expert policy is conditioned on an unobserved variable $z$ specifying its mode of behavior. This variable is sampled from a known discrete or continuous distribution $p(z)$ before each rollout. Formally, we consider the following generative process in multimodal settings:

$$z \sim p(z), s_0 \sim p_0, a_t \sim \pi_E(a_t|s_t, z), s_{t+1} \sim P(s_{t+1}|a_t, s_t).$$

3 Method

To enable imitation learning from diverse trajectories, we first introduce a novel generative model capable of learning arbitrary distributions. For an input data point $x \in \mathbb{R}^n$, we assume that a corresponding output data point $y \in \mathbb{R}^m$ is generated by a two-stage random process:

1. A latent variable $z$ (independent of $x$) is sampled from some prior distribution $p(z)$. The prior $p(z)$ can be a given discrete distribution over $K$ categories with probability masses $\pi_1 \ldots \pi_K$, or a continuous variable in $\mathbb{R}^d$ with an arbitrary density, e.g. $p(z) = \mathcal{N}(z; 0, I)$.

2. A value $y$ is drawn from a conditional distribution $p(y|z, x; f)$ of the following form:

$$p(y|z, x; f) = \mathcal{N}(y; f(z, x), \sigma^2 I), \tag{4}$$

where $f : \mathbb{R}^{n+d} \rightarrow \mathbb{R}^m$ is a function that maps $z$ and $x$ to the mean of the distribution, and $\sigma^2$ is a constant scalar that scales the identity covariance matrix.

New consider a dataset of $N$ i.i.d samples, $D = \{(x_i, y_i)\}_{i=1}^N$, generated by the above process. Our goal is to estimate the function $f$ within a family of parameterized functions $f_\theta : \mathbb{R}^{n+d} \rightarrow \mathbb{R}^m$, e.g. neural networks with weights $\theta$, by maximizing the log likelihood of the observed data:

$$\max_\theta \sum_{i=1}^N \log p(y_i|x_i; \theta) = \max_\theta \sum_{i=1}^N \log \int p(y_i|z, x_i)p(z)dz, \tag{5a}$$

$$= \max_\theta \sum_{i=1}^N \log \int \exp \left(-\frac{|f(z, x_i; \theta) - y_i|^2}{2\sigma^2} \right) p(z)dz, \tag{5b}$$

In this section, we study two cases of discrete and continuous latent variable and provide a unified algorithm to solve the optimization problem (5) in both cases.

3.1 Discrete Latent Variables

Expectation-maximization (EM) [20] is a general technique for finding maximum likelihood estimators in latent variable models. In Appendix A, we deploy the EM algorithm for the maximum likelihood problem of Equation (5), and obtain an iterative procedure that alternates between the following two steps:

1. **Expectation Step.** At iteration $t$, calculate the posterior probabilities,

$$r_{il}^t = p(z_i = l|(x_i, y_i); \theta^t, \{\pi_j^t\}_{j=1}^K). \tag{6}$$

2. **Maximization Step.** Update $\theta^{(t+1)}$ and the prior probabilities $\{\pi_j^{(t+1)}\}_{j=1}^K$ by maximizing the expectation value,

$$\sum_{k=1}^K \sum_{i=1}^N r_{ik}^t \log p((x_i, y_i), z_i = k | \theta, \{\pi_j\}_{j=1}^K). \tag{7}$$

3
We provide numerical experiments in Appendix A showing that the EM procedure above converges to good local minima. However, the speed of convergence is relatively slow and the storage requirements grow linearly in the number of data points. In addition, we show that the posterior probabilities, $r_{tl}$'s, converge to one-hot assignments. That is, for large enough $t$, $r_{tl} \rightarrow \gamma_{ik} \in \{0, 1\}$, where for any $i$ there exists a unique $j$ such that $\gamma_{ij} = 1$ and $\forall k: k \neq j$. This implies that at convergence, each data point pair $(x_i, y_i)$ is assigned a unique latent code $z_i = j$. This observation, along with analysis in Appendix A supporting the convergence of posterior probabilities to such binary extremes, motivate us to directly search over hard assignments of data points to latent codes.

For our numerical experiments, as reported in Appendix A, this strategy leads to faster convergence, and achieves better local maxima. Hence, we can propose the following combinatorial optimization problem for optimizing the model parameters:

$$\min_{\gamma_{ik}} \min_{\theta} \sum_{k=1}^{K} \sum_{i=1}^{N} \gamma_{ik} \|f(z_i = k; x_i; \theta) - y_i\|^2,$$

where $\gamma_{ik} \in \{0, 1\}$ is a binary indicator variable as defined earlier, and the $L_2$ error follows from the log probabilities in Equation (7). We shall consider a hard version of the EM algorithm to solve the optimization problem (8): (i) For a fixed $\theta$ greedily pick the best latent code that leads to the least fitting error for each data point. For each data point $i$, this is equivalent to determining the latent code, say $j$, with maximum posterior probability, $r_{ij}$, to 1 and setting the rest of $r_{il} = 0$; (ii) Optimize the model parameter $\theta$ with the corresponding latent code assignments. This hard assignment method is closely analogous to the derivation of the $K$-means algorithm from a corresponding soft EM in Gaussian mixture models [21]. See Appendix A for more details.

### 3.2 Continuous Latent Variables

For the continuous latent variable case, we follow a similar framework as in the discrete case: (i) For a fixed $\theta$ search among the samples of continuous latent space for the best code $z_i^*$ that leads to the least fitting error for each data point $(x_i, y_i)$; (ii) Optimize for $\theta$ by training the model for output $y_i$, and input $(x_i, z_i^*)$.

Finally, we shall make a few additional remarks about the continuous case and the resulting SOG algorithm:

**Relationship with other non-adversarial generative models.** Unlike conventional non-adversarial generative models such as the VAE, the SOG model does require an encoder. In VAE, the encoder is used to learn a posterior distribution $p(z|x_i, y_i; \theta)$, whose samples are used to train the decoder. In contrast, SOG trains the decoder by searching for a latent code $z$ that maximizes the likelihood $p(y|x, z; \theta)$. By using the basic marginal likelihood identity [22], one can show that this is equivalent to maximizing the ratio of the posterior to the prior, $p(z|x_i, y_i; \theta) / p(z)$. In Appendix B, we show that under reasonable assumptions, the continuous version of the SOG also leads to the maximization of the marginal log-likelihood of the data $p(Y|X; \theta)$.

**Sample complexity.** For the case of continuous latent variable, the number of latent code samples required in step 6 of Algorithm 1 can grow prohibitively large with the dimensionality of the latent space. However, this dimensionality is a design choice determined by the nature of the dataset. In the scope of imitation learning, which is the focus of this paper (see Section 3.4), low dimensions of the latent code are often sufficient and therefore Algorithm 2 is computationally tractable. Moreover, in Appendix D, we present an extension of Algorithm 1 that is computationally efficient for latent spaces with larger dimensions, and evaluate this extension with visual results.

**Self-organization and clustering in the latent code space.** In Appendix C we show analytical results that show how “self-organization” might emerge in Algorithm 1. That is, the latent codes corresponding to nearby data points get organized close to each other. In addition, visual results in Appendix D demonstrate that different regions of the latent space organize towards generating different modes of data. Hence the name of the method is justified.

### 3.3 The General SOG Model

Combining the ideas from Sections 3.1 and 3.2, we propose Algorithm 1 as a general procedure that applies to both the discrete and continuous latent codes.
Algorithm 1 Self-Organizing Generative Model (SOG)

1: Define: Loss function $\mathcal{L} (\hat{y}, y) := || \hat{y} - y ||^2$
2: for epoch = 1, 2, ... do
3: for $i = 1, 2, \ldots$ do
4: Sample a minibatch of $N_{\text{data}}$ data points $(x_j, y_j)$ from the dataset.
5: for $j = 1, 2, \ldots, N_{\text{data}}$ do
6: Sample $N_z$ latent codes $z_k \sim p(z)$.
7: Calculate $z_j := \arg \min_{z_j} \mathcal{L} (f_\theta(z_k, x_j), y_j)$.
8: end for
9: Calculate $\mathcal{L}_{\text{SOG}} = \sum_{j=1}^{N_{\text{data}}} \mathcal{L} (f_\theta(z_j, x_j), y_j)$ and its gradients w.r.t. $\theta$.
10: Update $f_\theta$ by stochastic gradient descent on $\theta$ to minimize $\mathcal{L}_{\text{SOG}}$.
11: end for
12: end for

3.4 SOG and SOG-GAIL for Imitation Learning

To encourage multimodal imitation learning, we introduce two approaches: (1) extending SOG to conditional settings for multimodal BC, and (2) combining conditional SOG with GAIL. We elaborate on each approach in the following.

Multimodal behavior cloning with SOG. Inspired by Algorithm 1, we aim to learn $p_\theta(a|s)$ for multimodal behavior cloning. Enforcing the latent codes to be shared across each trajectory, we obtain Algorithm 2.

Algorithm 2 SOG-BC: Multimodal Behavior Cloning with SOG

1: Define: Loss function $\mathcal{L} (\hat{a}, a) := || \hat{a} - a ||^2$
2: Input: Initial parameters of policy network, $\theta_0$; expert trajectories $\tau_E \sim \pi_E$
3: for $i = 0, 1, 2, \ldots$ do
4: Sample state-action pairs $\chi_E \sim \tau_E$ with the same batch size.
5: Sample $N_z$ latent codes $z_E^{(j)} \sim p(z)$ for each trajectory.
6: Calculate $z_E := \arg \min_{z_j} \mathbb{E}_{\tau_E} [\mathcal{L} (f_\theta(s, z_j), a)]$.
7: Calculate $\mathcal{L}_{\text{SOG}} = \mathbb{E}_{\tau_E} [\mathcal{L} (f_\theta(s, z_E), a)]$ and its gradients w.r.t. $\theta$.
8: Update $f_\theta$ by stochastic gradient descent on $\theta$ to minimize $\mathcal{L}_{\text{SOG}}$.
9: end for

Combination of behavior cloning and GAIL with mode separation using SOG. Motivated by [15], in which a weighted combination of the SOG loss and the GAIL surrogate loss is optimized, we introduce Algorithm 3. The SOG term encourages multimodal BC, while the GAIL term ensures robustness w.r.t. unseen states via exploration. We evaluate the effectiveness of this model in Section 4.

4 Experiments

In our experiments, we show that our method recovers and distinguishes all modes of behavior, and robustly replicates each mode with high fidelity. We evaluate our models, SOG-BC and SOG-GAIL, against two multimodal imitation learning baselines: InfoGAIL [8] and VAE-GAIL [9].

4.1 Experimental Setup

We first consider an experiment from [8] in which an agent can move freely at limited velocities in a 2D plane. The observed state at time $t$ is a concatenation of positions from time $t - 4$ to $t$. Expert demonstrations appear in three modes, each trying to produce a distinct circle-like trajectory. The expert tries to maintain a $(2\pi/100)$ rad/s angular velocity along the perimeter of the circle, however its displacement at each step incurs a 10%-magnitude 2D Gaussian noise from the environment.

In another series of experiments, we evaluate our model on several complex robotic locomotion tasks, borrowed from [23], which are simulated via the MuJoCo simulator [24]. These tasks include discrete-mode experiments HalfCheetah-Fwd-Back, Ant-Fwd-Back, Ant-Dir-6, and a continuous-mode experiment HalfCheetah-Vel. The name of the environments come from conventional MuJoCo environments. The suffix “Fwd-Back” indicates two modes of moving forward or backward, “Dir-6”
Algorithm 3 SOG-GAIL

1: **Input**: Initial parameters of policy and discriminator networks, \(\theta_0, w_0\); expert trajectories \(\tau_E \sim \pi_E\)
2: **for** \(i = 0, 1, 2, \ldots\) **do**
3: Sample a batch of latent codes \(z_i \sim p(z)\), and subsequently trajectories \(\tau_i \sim \pi_{\theta_i}(\cdot|z_i)\) with the latent code fixed across each trajectory.
4: Sample state-action pairs \(\chi_i \sim \tau_i\) and \(\chi_E \sim \tau_E\) with the same batch size.
5: Update the discriminator parameters from \(w_i\) to \(w_{i+1}\) with the gradient
   \[
   \hat{E}_{\tau_i} \left[ \nabla_w \log D_w(s, a) \right] + \hat{E}_{\tau_E} \left[ \nabla_w \log (1 - D_w(s, a)) \right].
   \]
6: Calculate the surrogate loss \(L_{\text{PPO}}\) using the PPO rule (equation 2) with the following objective
   \[
   \hat{E}_{\tau_i} \left[ \nabla_w \log D_w(s, a) \right] - \lambda H(\pi_{\theta_i}).
   \]
7: Calculate the SOG loss \(L_{\text{SOG}}\) per algorithm 2.
8: Take a policy step from \(\theta_i\) to \(\theta_{i+1}\) w.r.t the objective of \(L_{\text{PPO}} + \lambda S L_{\text{SOG}}\).
9: **end for**

indicates moving along six angles \(k\pi/3\), \(k = 0, 1, \ldots, 5\), and “Vel” indicates movement at a fixed horizontal velocity per trajectory that is sampled uniformly from the interval of 1.5 m/s to 3 m/s (as a reference, the cheetah’s torso length is around 1m). All these tasks have a horizon length of 200.

We use a few-shot reinforcement learning algorithm, Pearl [23], to efficiently train expert agents for the multimodal tasks above in order to generate expert trajectories.

In benchmarking the VAE-GAIL baseline, we observed that the VAE module performs well at embedding trajectories corresponding to each mode into distinct clusters in the latent space. However,

---

**Figure 1**: Discrete latent variables: robotic locomotion tasks. Trajectories of the imitated policy for three tasks. Respectively from top to bottom: number of modes are 3, 2, 6 (color coding corresponds to different latent codes at inference time); number of trajectories per latent code are 1, 3, 3; and horizon lengths are 1000, 200, 200. Both SOG-BC and SOG-GAIL achieve precise mode separation and behavior tracking of experts for all three experiments. In contrast, InfoGail and VAE-GAIL* (an ideal variant of VAE-GAIL we designed to make it work better for the discrete case; without the modification its performance is almost random) fail in separating the modes for Ant-Dir-6 (bottom row), and display significantly worse behavior imitation for all three experiments (See also Table 1).
when the GAIL module is conditioned on the learned embeddings, the resulting policies perform poorly. We attribute this problem to the spread of the latent codes within each cluster, which prohibits the policy from sufficiently exploring the entire spread of the latent codes within each cluster. To achieve better results with this model, we use the ground-truth mode labels and replace the trajectory embeddings with the average embedding among all trajectories that share the same ground truth label. This way, we can analyse the performance of a conditional GAIL given an ideal trajectory encoder. In Figure 1 and Table 1, we refer to this ideal variant as VAE-GAIL*. In fact, this highlights a major limitation of the VAE-GAIL model in imitating policies with only few discrete modes.

We run each experiment with four random seeds and present the results for the best training iteration in terms of expected sum of rewards. Further experimental details are described in Appendix E.

4.2 Evaluation

Visualization. In Figure 1, we illustrate the generated trajectories for the Circles experiment, and 2D locomotion trajectories for Ant-Fwd-Back and Ant-Dir-6 experiments. The plots of SOG-GAIL correspond to the choice of $\lambda_S = 1$ in Algorithm 3. Across all plots, we observe that both SOG-BC and SOG-GAIL successfully learn all the modes. Also, it can be seen that SOG-BC performs slightly more accurately compared to SOG-GAIL. We discuss the differences between SOG-BC and SOG-GAIL models in more details in a later paragraph, where we address their relative robustness to different types of perturbations and scenarios. For the continuous experiment HalfCheetahVel, in Figure 2 we visualize the spread of instantaneous horizontal velocities as an input control variable varies. In SOG-BC and InfoGAIL, this control variable is the 1-D latent code used to train the model. Since VAE-GAIL works well only with high dimensional latent variables, one cannot directly visualize the correspondence between latent codes and the model generated horizontal velocities. Instead, we utilize the fact that each expert trajectory is generated with a target velocity. Thus, if the embedding of expert trajectories (i.e. corresponding latent codes) is fed as input to the learned policy, the generated trajectories will produce very similar velocities as the expert trajectories. Thus, for VAE-GAIL we study the variations in the generated velocities as a function of the desired velocities (i.e. the target set by the experts). Compared to the baselines, we observe that SOG-BC can better associate the latent variable to the goal velocities in the range $[1.5, 3]$, with less uncertainty (narrower error band). Since SOG-GAIL for $\lambda_S = 1$ gives a plot similar to that of SOG-BC, we found it redundant to include SOG-GAIL.

![Figure 2: Continuous latent variable: the HalfCheetahVel experiment.](image)

Metrics. After training the imitated policy, one can use the ground-truth reward functions to evaluate the collected rewards in the best iteration of different models. For the Circles experiments, applying a Gaussian kernel to the distance of the agent from the perimeter of each reference trajectory, we obtain a reward between 0 and 1 at each step. For Mujoco environments, we use the original definition of the rewards for each mode introduced in [23]. To find the corresponding mode-specific reward for each latent code, we select a correspondence of latent codes to actual modes that gives the best expected sum of rewards. For HalfCheetahVel experiment, we estimate the mutual information,
Figure 3: Robustness of SOG-GAIL vs SOG-BC: Ant-Dir-6 experiment. To simulate unseen situations, we create a task, as shown in (a), where the locomotion direction of the ant agent is abruptly changed (corresponding to different expert modes). In (b), (c) three trajectories generated are shown in different colors. We observe that SOG-GAIL performs the desired task flawlessly, while the ant agent trained with SOG-BC often topples over and fails the task.

Table 1: Mean and standard deviation of rewards collected in 100 rollouts of each algorithm

<table>
<thead>
<tr>
<th>Data set</th>
<th>SOG-BC</th>
<th>SOG-GAIL ($\lambda_S = 1$)</th>
<th>InfoGAIL</th>
<th>VAE-GAIL*</th>
<th>Expert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circles</td>
<td>992.1 ± 1.0</td>
<td>985.9 ± 10.8</td>
<td>766.0 ± 67.9</td>
<td>912.3 ± 8.6</td>
<td>908.3 ± 0.1</td>
</tr>
<tr>
<td>Ant-Fwd-Back</td>
<td>1165.2 ± 32.1</td>
<td>1101.0 ± 61.7</td>
<td>220.6 ± 296.3</td>
<td>−385.3 ± 67.0</td>
<td>1068.7 ± 109.7</td>
</tr>
<tr>
<td>Ant-Dir-6</td>
<td>1073.2 ± 206.6</td>
<td>1023.2 ± 166.1</td>
<td>−14.5 ± 87.6</td>
<td>−572.9 ± 62.9</td>
<td>1031.7 ± 253.7</td>
</tr>
<tr>
<td>HalfCheetah-Fwd-Back</td>
<td>221.6 ± 957.3</td>
<td>1532.6 ± 148.5</td>
<td>484.2 ± 919.4</td>
<td>84.0 ± 152.2</td>
<td>1686.0 ± 135.4</td>
</tr>
</tbody>
</table>

Robustness. We design two experiments to show how combining SOG with GAIL helps towards making the learned policy robust to unseen states. In the Ant-Dir-6 experiment, the locomotion direction is unchanged in each expert trajectory. We change the angle of movement to all six directions, which creates unseen circumstances. As illustrated in Figure 3, in maneuvers with acute angles, the simulated ant trained with SOG-BC topples over, and the episode terminates. However, the agent trained with SOG-GAIL successfully switches between all 6 directions. In Figure 4a, we observe that except for HalfCheetah-Fwd-Back, increasing the coefficient $\lambda_S$ from Algorithm 3 improves scores in Table 1 across all experiments. We now perturb each learned policy by taking random actions with 20% probability. We observe that under this perturbation, several of the experiments perform worse at higher values of $\lambda_S$. This indicates that combination of SOG with GAIL is more robust to perturbations.

Table 2: Mutual information between the latent variable (mean velocity of the embedded trajectory in the case of VAE-GAIL) and the generated velocity.

<table>
<thead>
<tr>
<th>Data set</th>
<th>SOG-BC</th>
<th>SOG-GAIL ($\lambda_S = 1$)</th>
<th>InfoGAIL</th>
<th>VAE-GAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>HalfCheetah-Vel</td>
<td>1.584</td>
<td>1.431</td>
<td>0.145</td>
<td>0.750</td>
</tr>
</tbody>
</table>

5 Concluding Remarks

We introduce a generative model that enables imitation learning of diverse behaviors when expert demonstrations follow a mixture of behaviors. We provide theoretical analysis that our model is capable of optimizing the maximum likelihood objective of the conditional distribution between input and output variables, when the underlying relationship depends on an unobserved variable. This model shows better performance for both cases of discrete and continuous latent variables. Across multiple complex experiments, we show that our model precisely reconstructs expert trajectories, while simultaneously distinguishing the underlying modes. We also demonstrate that a natural combination of our generative model with adversarial training boosts the robustness of the imitated policies to unseen states.
References


