Influencing Towards Stable Multi-Agent Interactions

Woodrow Z. Wang, Andy Shih, Annie Xie, Dorsa Sadigh
Department of Computer Science, Stanford University
{wwang153, andyshih, anniexie, dorsa}@stanford.edu

Abstract: Learning in multi-agent environments is difficult due to the non-stationarity introduced by an opponent's or partner's changing behaviors. Instead of reactively adapting to the *other agent's* (opponent or partner) behavior, we propose an algorithm to proactively influence the other agent's strategy to *stabilize* – which can restrain the non-stationarity caused by the other agent. We learn a low-dimensional latent representation of the other agent's strategy and the dynamics of how the latent strategy evolves with respect to our robot's behavior. With this learned dynamics model, we can define an unsupervised stability reward to train our robot to deliberately influence the other agent to stabilize towards a single strategy. We demonstrate the effectiveness of stabilizing in improving efficiency of maximizing the task reward in a variety of simulated environments, including autonomous driving, emergent communication, and robotic manipulation. We show qualitative results on our website.

Keywords: multi-agent interactions, human-robot interaction, non-stationarity

1 Introduction

Reinforcement learning has shown impressive performance in recent years [1, 2], but much of the focus has been on single-agent environments. Many real-world scenarios, on the other hand, involve coordinating with multiple agents whose behavior may change over time, making the task non-stationary. For example, when playing volleyball, our teammate may play more aggressively over time if they are hitting the ball well, so we may need to step aside and set the ball for them more often. Many techniques in multi-agent reinforcement learning (MARL) have been proposed for dealing with such non-stationarity in the other agent's strategies [3].

One set of approaches reactively adapts to the other agent's change in strategy, for example by simply forgetting past experiences and training on more recent data [4, 5, 6, 7, 8]. Other works try to learn a model of the other agent and take into account how the robot's actions affect the other agent's strategies over time [9, 10, 11, 12, 13]. With knowledge of the other agent's strategy dynamics, we can take actions to influence the other agent in a way that maximizes our long-term reward [10, 13].

Although influencing other agents to maximize reward is a useful approach in principle, it can be challenging in practice. Indeed, if we have access to a faithful model of the other agent's strategy dynamics, then we should choose actions that carefully nudge their high-level strategy to maximize our long-term reward. However, learning and leveraging such a model of the other agent's strategy dynamics can be impractical. For one, even just estimating the other agent's current strategy is already difficult with limited data, let alone learning the dynamics between strategies. On top of that, the robot may have to figure out a best response for a large number of opponent strategies, making the full learning process extremely sample inefficient.

Instead, we should take into account the difficulty of learning for our ego agent. [1] That is, we would still like to influence the other agent in a useful way, but in a manner that minimizes the burden of the ego agent's learning algorithm. *Our key insight is that we can stabilize the other agent's strategy in a multi-agent environment to reduce non-stationarity and facilitate learning a best response to the other agent.* Our approach is a specific form of influencing – learning how to stabilize – that

[1] We refer to the other agent interchangeably as the opponent, regardless of whether they are competitive or cooperative [3]. Also, we interchangeably refer to our robot as the ego agent.

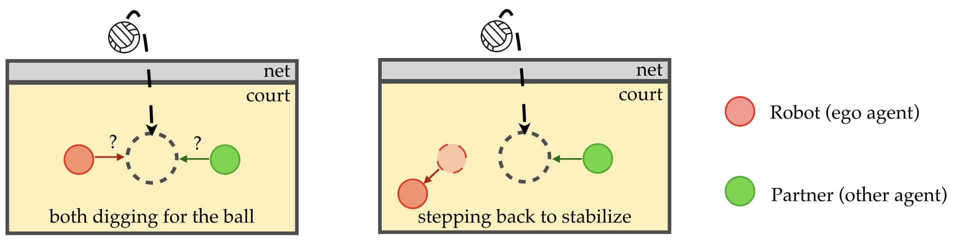

Figure 1: Coordination with a partner can be critical to succeeding in a task. In this volleyball game, the red and green agents both can either dig for the ball or step back for their partner. (Left) A partner that indecisively switches between digging and staying back can be extremely difficult to coordinate with, potentially costing the match. (Right) By deliberately stepping back, the red agent can stabilize the partner's strategy to dig for the ball, developing a convention that allows for easier learning of how to win the volleyball match.

aims to reduce the non-stationarity of the other agent, and in turn makes learning easier for the ego agent. Intuitively, stabilizing the other agent's strategy allows us to avoid learning about complex high-level strategy dynamics, and focus our efforts on maximizing reward against a fixed strategy.

As an example of stabilizing the other agent, imagine again that we are playing beach volleyball with a teammate. When the ball comes perfectly in between our teammate and us, it is not obvious which of us should dig for the ball (see Fig. 1 (left)). In principle, it is possible to carefully model the dynamics of how our teammate's strategy changes based on our actions and precisely predict who should get the next ball that falls in between us. However, it can take many interactions before we learn a good model of our opponents, and constantly switching roles between digging for the ball and staying back can make learning difficult. Instead, we can stabilize our teammate by making our intentions clear: we actively take a step away from the ball so our teammate knows to dig for the ball, as in Fig. 1 (right). This simplifies the learning process by reducing the need to learn complicated strategy dynamics, and making the task more stationary.

In particular, our method combines the robot's task reward with a stabilizing reward, where stability is measured with respect to the other agent's strategies learned in an unsupervised way. The task reward pushes the robot to do well in the environment, while the stabilizing reward aims to make the learning process easy and sample efficient. We demonstrate the strength of our approach in 7 different tasks, including autonomous driving, an emergent communication task, and simulated robot environments.

Our main contributions in this paper are as follows:

- We propose augmenting our robot's reward function with a stability reward in order to explicitly encourage the robot to influence the other agent towards a stable strategy.
- We demonstrate that the stability reward can be defined as an unsupervised reward using learned representations of the other agent's strategy.
- We perform experiments in simulated environments demonstrating that stabilizing the other agent's strategy improves our robot's ability to coordinate and maximize the task reward.

2 Related Work

Multi-Agent Reinforcement Learning. Learning in multi-agent environments can be extremely difficult when the other agents have non-stationary behavior [14, 15, 16]. Hernandez-Leal et al. [3] provide a taxonomy of existing methods that attempt to deal with this non-stationarity issue in multi-agent environments. Some works seek to forget about previous experiences with old opponent strategies and improve adaptation by only training on more recent data [4, 5, 6, 7, 8, 17]. Further, centralized training has shown promise in multi-agent environments [18, 19]. Some works design explicit communication channels in which agents can share their policy parameters or gradient updates [20, 21]. However, these methods require that there exists some centralized method in which agents can communicate or share parameters. We are mainly interested in the decentralized setting where we train an ego agent to interact with other agents that we do not have control over.

Opponent Modeling. Instead of ignoring the opponent's changing strategies, prior opponent modeling works design agents that estimate their opponent's policies or parameter updates [22, 9, 23, 11, 12, 24, 25] in a potentially recursive manner. Peer-rewarding has also emerged as a method of encouraging cooperative behavior [26, 27, 16], but requires the assumption that an agent's reward function can be externally modified. Several works recognize that the ego agent's policy can directly induce change in the opponent's policies [13, 28, 29, 30]. In [10], the ego agent learns not only a representation of the opponents' strategies, but also learns a dynamics model for how the opponent's strategies change conditioned on the ego agent's behavior.

Representation Learning of Latent Strategies. Representation learning is critical to the tractability of opponent modeling. It may be computationally inefficient to learn all opponent intentions or explicitly learn the parameters of the opponent's policies. Instead, we recognize that there is often some underlying structure in the opponent policy space, which can be captured by a low-dimensional latent strategy. Prior works learn latent context variables to help distinguish between separate tasks, which could correspond to different opponents [31, 32, 33, 34, 35]. Representation learning has been shown to be useful in developing a dynamics model of the opponent's policy in multi-agent environments [36, 10]. In our work, we show that it can be beneficial to cluster latent strategy representations into a discrete latent space [37, 38] to identify shared structure among similar strategies.

Stability and Surprise in Reinforcement Learning. To mitigate the difficulty of learning in a non-stationary environment, we seek to stabilize the opponent's strategy – or minimize the surprise in the opponent's strategy. Recent works in reinforcement learning seek to do the opposite of designing maximum entropy methods to improve exploration [39, 40, 41]. Stability in the opponent's strategy can reduce the non-stationarity in the environment and improve efficiency of learning a best response [42]. Surprise minimizing reinforcement learning (SMiRL) trains an agent to seek predictable and repeatable states to counteract the environment's entropy [43]. SMiRL considers a long horizon notion of stability in which the state should be of high likelihood with respect to the distribution of states encountered so far. Our work is similar in principle to SMiRL; however, we consider a pairwise notion of stability between consecutive latent strategies, as we recognize that an opponent that unpredictably switches between a few strategies is highly non-stationary, yet each individual strategy can be high likelihood with respect to the overall distribution of previously seen strategies.

3 Problem Statement

We consider two-player multi-agent settings, where there is an ego agent that we have control over and another agent that we do not have direct control over. Since we only control the ego agent, we can consider our problem formulation in terms of a single-agent Markov decision process where the opponent policy is absorbed as part of the environment transitions and rewards. Non-stationarity in the opponent policy then corresponds to non-stationarity in the environment transitions and rewards.

Hidden Strategies. The agents repeatedly interact with each other across multiple interactions, with each interaction lasting multiple timesteps. We assume that the opponent policy z is fixed within each interaction but can change across interactions. We consider environments with no hidden states, so that if the ego agent can observe the exact opponent strategy, then we can formulate the ego agent's task as a fully observable MDP. However, since the ego agent does not directly observe the opponent's strategy, we consider a Hidden Parameter MDP (HiP-MDP).

Hidden Parameter Markov Decision Process. We formulate the interactions as a variant of a HiP-MDP, which is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{Z}, \mathcal{T}, \mathcal{T}^z, \mathcal{R}, H, \gamma \rangle$ [44], which includes an additional $\mathcal{T}^z$ that governs the dynamics of the opponent's strategies. We let $z^j \in \mathcal{Z}$ be the opponent strategy for the j-th interaction. At each timestep t, the ego agent is in state $s_t \in \mathcal{S}$. The ego agent takes action $a_t \in \mathcal{A}$ and the environment transitions to s_{t+1} with probability $\mathcal{T}(s_{t+1}|s_t, a_t, z^j)$. The ego agent receives the task reward $\mathcal{R}_{\text{task}}(s_t, a_t, z^j)$. The interaction ends after H timesteps and γ is the discount factor. Both the transition and reward function are unknown. During interaction j, the ego agent observes the trajectory $\tau^j = \{(s_1, a_1, R_{\text{task}}(s_1, a_1, z^j)), \ldots (s_H, a_H, R_{\text{task}}(s_H, a_H, z^j))\}$. Notice that the ego agent does not observe the opponent's strategy z^j in the trajectory, which makes learning the value of an observation and action difficult – the reward of an observation and action can change significantly as the opponent's strategy changes across interactions.

We consider the setting where the opponent's change in strategy is governed by the trajectory of the previous interaction. At the end of interaction j, the opponent's strategy transitions to z^{j+1} based on strategy dynamics $\mathcal{T}^z(z^{j+1}|z^j,\tau^j)$. Namely, the ego agent's behavior during interaction j can directly influence the opponent's strategy for interaction $j+1$. Next, we define a notion of stability between interactions j and $j+1$ as follows:

Definition 1 *Let $z^j, z^{j+1} \in \mathcal{Z}$ be two opponent strategies. The strategies z^j, z^{j+1} are **pairwise ϵ-stable** with respect to a metric space $(\mathcal{Z}, d)$ if and only if $d(z_{j+1}, z_j) < \epsilon$.*

Stabilizing Opponent Strategies by Influencing Across Repeated Interactions. Over repeated interactions, the ego agent explores HiP-MDP $\mathcal{M}$ over a sequence of opponent strategies $(z^1, z^2, \ldots)$. The ego agent's objective is to maximize the cumulative discounted sum of rewards over all interactions, shown in Eq. 1. The ego agent's ability to maximize the reward within each interaction is heavily dependent on their ability to estimate the opponent strategy z^j for that interaction. Without knowing the opponent's strategy, the ego agent will have significant difficulty taking an action that is a best response to the opponent's actions. It can be prohibitive to learn a model of all the opponent's strategies and form a general policy for the ego agent that can react to any opponent strategy; however, if the ego agent can stabilize the opponent strategy to a fixed point, i.e., $\exists z^k$ such that $\mathcal{T}^z(z^{j+1}|z^j,\tau^j) > 0$ if and only if $d(z^j, z^{j+1}) < \epsilon, \forall j \in \{k \ldots N\}$ (where N is the total number of interactions), then the ego agent can more easily learn a best response to the opponent strategy in a stationary environment.

With z fixed across interactions, the HiP-MDP is reduced to a fully observable MDP parameterized by z. At a high level, we are encouraging the agent to learn how to explore the HiP-MDP in a subset of the state space, where the hidden opponent strategy is kept stable between interactions. Thus, the ego agent is effectively learning in a fully observable MDP, given that the fixed – but unknown – opponent strategy z is not changing. In terms of the exploration-exploitation tradeoff, stabilizing the opponent strategy would reduce the ego agent's required exploration in order to find a best response to the opponent's fixed strategy.

We note that it is not always easy or possible to *stabilize* the opponent's strategy. Stabilizing requires that the dynamics of the opponent's strategy are defined such that the ego agent can form an interaction τ^j where $\mathcal{T}^z(z^{j+1}|z^j,\tau^j) > 0$ if and only if z^j and z^{j+1} are pairwise ϵ-stable. In general, as long as there is a fixed point z^k where the optimal value function satisfies $V^*(s_1, z^k) \geq V^*(s_1, z)$ for any other $z \in \mathcal{Z}$, then stabilizing should not hinder the task reward in the long run.

4 Stable Influencing of Latent Intent (SILI)

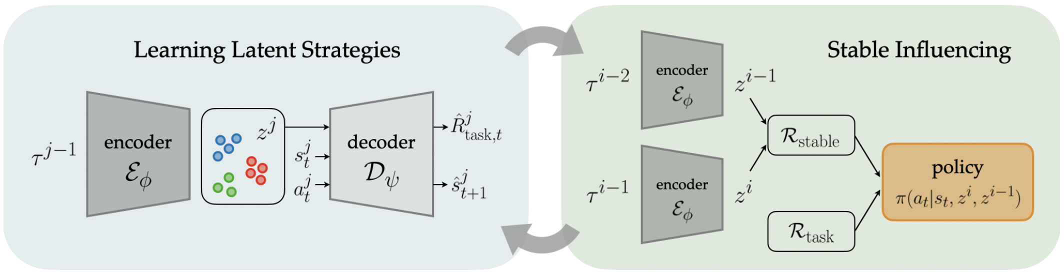
Learning Latent Strategies
τ^{j-1}
encoder $\mathcal{E}_\phi$
z^j
s^j_t
a^j_t
decoder $\mathcal{D}_\psi$
$\hat{R}^j_{\text{task},t}$
$\hat{s}^j_{t+1}$
Stable Influencing
τ^{i-2}
encoder $\mathcal{E}_\phi$
z^{i-1}
τ^{i-1}
encoder $\mathcal{E}_\phi$
z^i
$\mathcal{R}_{\text{stable}}$
$\mathcal{R}_{\text{task}}$
policy $\pi(a_t|s_t, z^i, z^{i-1})$

Figure 2: Stable Influencing of Latent Intent (SILI). (Left) We learn a dynamics model of the opponent's strategies conditioned on the ego agent's past trajectory. The latent strategy is learned in an unsupervised way by jointly learning a decoder to predict the state transitions and task rewards. (Right) We combine a stability reward with the task reward to train our policy using RL. The stability reward is unsupervised and defined by minimizing the pairwise distance between the previous two predicted latent strategies.

In this section, we present our approach of influencing the latent strategy of our opponent towards a stable strategy. At a high level, the ego agent learns an encoder that predicts the opponent's next latent strategy given the previous trajectory. The ego agent then conditions its policy for the next interaction on the predicted latent strategies of the opponent. But, since we never observe the ground truth latent strategies, we must learn the latent strategy representations in an unsupervised way. To do so, we follow the setup in [10], where we jointly learn a decoder that reconstructs the task rewards and transitions for the next interaction using the predicted latent strategy. Using this predicted latent strategy, we can measure the stability of the opponent's latent strategy, and optimize for a weighted combination of the task reward and a stability reward.

Learned Latent Strategies. First, we describe the representation learning component of our method shown in Fig. 2 (left). Recall that we assume that the latent strategy dynamics follow $\mathcal{T}^z(z^{j+1}|z^j, \tau^j)$. We can approximate these latent strategy dynamics by learning an encoder $\mathcal{E}_\phi$ that maps trajectories τ^{j-1} to a latent representation z^j of dimension k (chosen as a hyperparameter). Here, we model the encoder as $f(z^j|\tau^{j-1})$ instead of $f(z^j|z^{j-1}, \tau^{j-1})$, as in [10]. This simplification, which often works well in practice, assumes that the previous trajectory τ_j already has enough information to reconstruct z^{j+1}.

Although we do not observe the ground truth latent strategies, we can still learn them in an unsupervised way, since we know that they should be predictive of the task reward and transitions. Thus, we jointly train a decoder that predicts the task reward and transitions from the encoded latent representation. In particular, for interaction j and timestep t, the decoder $\mathcal{D}_\psi$ takes as input states and actions s_t^j, a_t^j as well as the latent strategy embedding z^j, and reconstructs the next state s_{t+1}^j and task reward r_t^j. To train the encoder and decoder, we use the following maximum-likelihood objective: $\max\limits_{\phi,\psi} \sum_{j=2}^N \sum_{t=1}^H \log p_{\phi,\psi}(s_{t+1}^j, r_t^j | s_t^j, a_t^j, \tau^{j-1})$.

Reinforcement Learning with Stable Latent Strategies. As the opponent's strategy changes, the ego agent's policy should change in response. Naturally, we should condition the ego agent's policy on the opponent's latent strategies. When considering the goal of keeping the opponent's strategy pairwise stable, it is important that the ego agent conditions their policy on at least the most recent two predicted latent strategies since the stability reward depends on the distance between the consecutive latent strategies, as shown in Fig. 2 (right). Concretely, the ego agent's policy is defined as $\pi_\theta(a_t|s_t, z^j, z^{j-1})$, where $z^j = \mathcal{E}_\phi(\tau^{j-1})$.

We have defined the environment reward to be $\mathcal{R}_{\text{task}}$. We then define the stability reward to be the distance between consecutive predicted opponent strategies, where d can be any generic distance metric over the latent space $\mathcal{Z}$ as follows: $\mathcal{R}_{\text{stable}} = -d(z^j, z^{j-1})$.

Here, we abuse notation and use z^j to refer to the prediction of the opponent's strategy since we never observe the ground truth. Finally, our new reward function is defined as a weighted average of these objectives, parameterized by stability weight β: $\mathcal{R}_{\text{total}} = (1-\beta)\cdot\mathcal{R}_{\text{task}} + \beta\cdot\mathcal{R}_{\text{stable}}$. With $\beta = 1$, the ego agent's entire objective is to stabilize the opponent strategy, and with $\beta = 0$, the ego agent greedily maximizes the environment reward.

Optimizing Rewards Across Interactions. Our ego agent seeks to influence the opponent towards desirable latent strategies. With the encoder $\mathcal{E}_\phi$, the ego agent can deliberately take actions in trajectory τ^j in order to influence the opponent to a stable z^{j+1}.

To learn how to influence across repeated interactions, we train our ego agent's policy π_θ to maximize rewards across the repeated interactions

$$\max_\theta \sum_{j=1}^{\infty} \gamma^j \mathbb{E}_{z^j, z^{j-1}, \tau^j \sim \rho_{\pi_\theta}^j} \left[\sum_{t=1}^{H} \mathcal{R}_{\text{total}}(s, a, z^j, z^{j-1}) \right], \tag{1}$$

where $\rho_{\pi_\theta}^j$ is the trajectory distribution induced by π_θ under the transition function $\mathcal{T}(s_{t+1}|s_t, a_t, z^j)$. With the total reward including the stability reward, the ego agent learns to take interactions such that the opponent follows a stable latent strategy so the ego agent can easily learn a best response to in turn maximize the task reward.

Modeling Latent Strategies. We examine three methods to model and measure distances between the changing latent strategies: continuous latent variables, discrete latent variables, and partial observations of the opponent's strategy. The ego agent never directly observes the opponent's strategy.

1) Continuous Latent Variables. In the most general case, the latent strategy space is allowed to be continuous. This enables the latent strategy space to represent an infinite number of possible opponent strategies. With the continuous latent variables $\mathcal{Z}$, we use the Euclidean norm for the stability reward: $\mathcal{R}_{\text{stable}} = -d(z^j, z^{j-1}) = -||z^j - z^{j-1}||_2$

2) Discrete Latent Variables. In this setting, we constrain the latent strategy space to be discrete categorical variables by using the Gumbel-Softmax distribution [37, 38]. Intuitively, we cluster the continuous latent variable representations to filter the noise of measuring distance in the latent space. Further details of the discretization method can be found in Appendix A. With these categorical latent variables $\mathcal{Z}$, we use the discrete distance metric for the stability reward:

$$\mathcal{R}_{\text{stable}} = -d(z^j, z^{j-1}) = \begin{cases} -1 & z^j \neq z^{j-1} \\ 0 & z^j = z^{j-1} \end{cases} \tag{2}$$

We found empirically that discrete latent variables improve the consistency of the stability reward signal compared to continuous representations, despite the decrease in expressivity and granularity provided by the distance metric. Intuitively, it is easier to identify whether or not a strategy is similar enough to be clustered with another strategy than it is to exactly identify the distance between predicted strategies in a latent space.

3) Partial Observations of Changing Strategy. In this setting, we assume that the ego agent is able to observe if their opponent's strategy changes between timesteps, so no representation learning is necessary to stabilize. We shed the assumption that the latent strategies are fixed within an interaction, so we denote the latent strategy at a timestep t as z_t. Thus, the stabilizing reward can be defined as $\mathcal{R}_{\text{stable}} = c_t = -\mathbb{1}[z_t \neq z_{t-1}]$, which indicates whether the opponent strategy has changed. This corresponds to settings where it may be easy to estimate when an opponent's strategy has changed, but it may be difficult to estimate their exact strategy [12].

5 Experiments

We perform experiments in a range of environments across different latent strategy space settings: continuous latent variables, discrete latent variables, and partial strategy observations. We compare our approach, Stable Influencing of Latent Intent (SILI) to various multi-agent learning baselines with implementation details described in Appendix A:

Oracle. Oracle is given the true opponent's strategy as an observation.

Learning and Influencing Latent Intent (LILI) [10]. LILI combines soft actor-critic (SAC) [45] with a representation learning component to model the opponent's strategy dynamics. LILI does not explicitly optimize for stabilizing the opponent's strategy.

Surprise Minimizing RL (SMiRL) [43]. SMiRL optimizes for a different notion of stability, where the reward is proportional to the likelihood of a state given the history of previously visited states.

Stable. We refer to Stable as the variant of SILI with $\beta = 1$, only optimizing for stability.

SAC [45]. SAC does not model the opponent's strategy and is the base RL algorithm for SILI.

5.1 Simulated Environments

The environments and trajectories of stabilizing behavior are shown in the left column of Fig. 3. Across all figures, the ego agent's trajectory is drawn in blue, and grey corresponds to the opponent.

Circle Point Mass. In this environment, the ego agent is trying to get as close to the opponent as possible in a 2D plane, inspired by pursuit-evasion games [46]. The opponent moves between locations along the circumference of a circle. The ego agent never observes the true location (strategy) of the opponent. If the ego agent ends an interaction inside the circle, the opponent jumps counterclockwise to the next target location. If the ego agent ends an interaction outside the circle, the opponent stays at the red target location for the next interaction. We examine four variants: Circle (3 Goals), Circle (8 Goals), Circle (Continuous), Circle (Unequal). In Circle (Continuous), there are infinite possible opponent strategies, so we use continuous latent variables. In Circle (Unequal), the ego agent begins closer to two unstable opponent goals, but there is a stable goal farther away.

Driving. A fast ego agent is attempting to pass a slow opponent driver [10]. There are 3 lanes and a road hazard upcoming in the center lane, so both the ego agent and opponent need to merge to a new lane. If the ego agent merges to the left lane before the red line (giving the opponent enough reaction time), then the opponent merges to the right lane during the next interaction, understanding the convention of faster vehicles passing on the left. Otherwise, the opponent will aggressively try to cut off the ego agent by merging into the lane that the ego agent previously passed in.

Sawyer-Reach. The opponent can choose between three goals on a table with their intent hidden from the robot. The ego agent is the Sawyer robot that is trying to move their end effector as close as possible to the opponent's chosen goal in 3D space without ever directly observing the opponent's strategy [47]. If the end-effector ends the interaction above a fixed plane on the z-axis, the opponent's strategy stays fixed. Otherwise, the opponent's strategy changes. Semantically, we consider a robot server trying to place food on a dish for a human and needing to move its arm away from the dish by the end of the interaction to avoid intimidating the human retrieving the food.

Detour Speaker-Listener. In this environment, the agents must communicate to reach a goal, which has been a popular setting to explore emergent communication [18]. The speaker (opponent) does not move and observes the true goal of the listener (ego agent). The ego agent cannot speak, but must navigate to the correct goal. The speaker utters a message to refer to the goal, which the ego agent then observes. If the ego agent goes near the speaker (within some radius), the speaker follows the same communication strategy during the next interaction. Otherwise, the speaker chooses a random new strategy, or mapping of goal landmarks to communications. Critically, in this environment, the opponent's strategy is the communication strategy, not the true goal location.

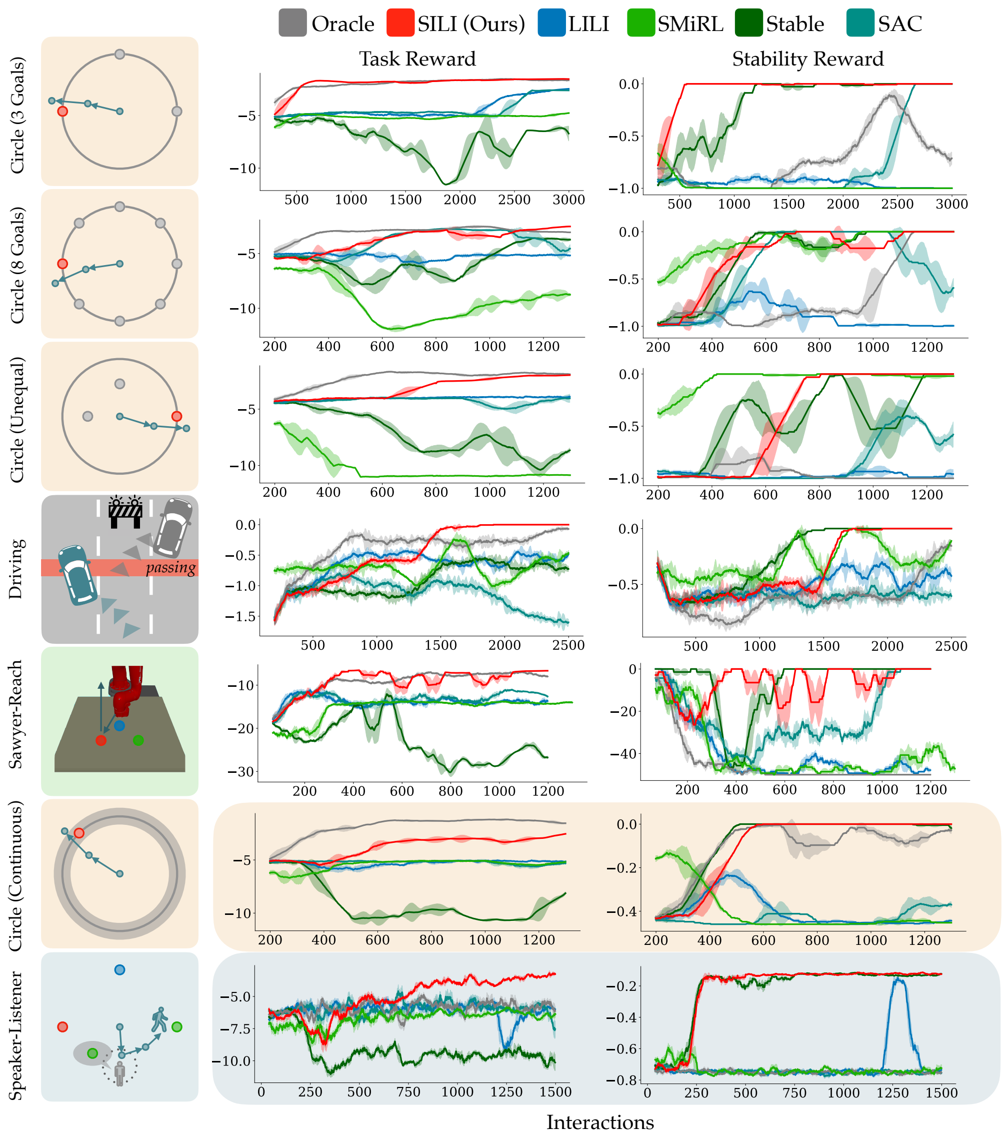

Figure 3: Task and stability rewards across all environments. SILI achieves comparable performance to Oracle. Compared to the other baselines, SILI significantly outperforms them in task reward (left plots) by learning to stabilize the opponent's strategy (right plots).

5.2 Results

Across all environments, without observations of the opponent's strategy, SILI achieves a comparable task reward to Oracle, as seen in Fig. 3. Further analysis is available in the Appendix and at `https://sites.google.com/view/stable-marl/` with qualitative analysis of trajectories.

Continuous Latent Variables. We perform an experiment with continuous latent variables in the Circle (Continuous) environment (row 6, Fig. 3). Both SILI and Oracle learn to stabilize in this environment, although Oracle incurs various dips in stability reward at little to no cost to the task reward because Oracle knows the true opponent's strategy. While these results are promising, we observe in other environments that continuous latent variables and a smooth distance metric over the latent space lead to more noisy rewards, which hinder the ego agent's ability to learn effectively.

Discrete Latent Variables. We further show our results in a variety of environments with discrete latent variables. In Circle (3 Goals), SILI achieves the highest reward along with Oracle, but Oracle does not require stabilizing the opponent's strategy (row 1, Fig. 3). In Circle (8 Goals), we demonstrate that with more opponent strategies, SILI still learns to stabilize and outperforms the baselines (row 2, Fig. 3), where we set the latent dimension hyperparameter to be an overestimate of the true number of opponent strategies. In Circle (Unequal), SILI avoids trying to optimize for nearby, unstable opponent strategies, whereas the other baselines do not learn to stabilize since they greedily try to track the nearby, unstable opponent strategies (row 3, Fig. 3). In Driving, the task reward is based on the number of collisions with the opponent, and our method is the only one that learns to safely avoid all collisions (row 4, Fig. 3). In Sawyer-Reach, SILI and Oracle achieve comparable performance, but again, Oracle does not need to learn to stabilize (row 5, Fig. 3).

Partial Observations of Changing Strategy. To test our method in a setting without representation learning, the ego agent observes whether the opponent's strategy has changed between timesteps. We perform this experiment in the Detour Speaker-Listener environment, where the assumption that the opponent's strategy is kept fixed throughout an interaction is broken (row 7, Fig. 3).

SILI learns to stabilize and significantly outperforms all other baselines in task reward, including Oracle. In this environment, although Oracle can observe the true opponent strategy, the Oracle still needs to learn the association of the opponent's strategy and opponent's communication with the true goal. Further, the communication and strategy can change frequently between timesteps while the true goal stays constant, making learning difficult. The baseline Stable learns to stabilize but does not optimize for the task reward at all. The rest of the baselines learn to move towards the centroid of the goal locations since they cannot accurately decipher the opponent's strategy.

Notice that around interaction 250, SILI learns to stabilize, which requires moving in a detour away from the true goals, but SILI maintains the stability and learns to maximize task reward well. On the contrary, LILI temporarily learns to stabilize at around interaction 1250, but incurs a corresponding decrease in task reward, which discourages LILI from maintaining the stable opponent strategy.

6 Discussion

Summary. We propose a framework for learning in multi-agent environments that learns the dynamics of the opponent's strategies in a latent space. We then leverage this latent dynamics model to design an unsupervised stability reward, which can be augmented with the task reward to explicitly encourage the ego agent to learn how to stabilize and then more easily maximize the task reward.

Limitations and Future Work. In this work, we define a notion of pairwise stability that considers consecutive changes in the strategy. However, it may be interesting to consider smoother, longer horizon stability metrics, such as average change over a window of time. We compare our method to SMiRL, which optimizes for a probabilistic notion of state stability relative to states in an experience replay buffer, but show that SMiRL often does not learn to stabilize in a consistent manner. We plan to compare these different metrics of stability and their effects in future work. In addition, our experiments were mainly done in simulation. In the future, we plan to consider robot-robot and human-robot interactions, where we are required to learn much more complex dynamics that capture the non-stationarity of humans who adapt their strategies over repeated interactions. Stabilizing a human's strategy can greatly improve the efficiency in which a robot learns the best way to perform a task, but the exact method in which to stabilize human behavior should be further explored.

Acknowledgments

We would like to acknowledge NSF grants #1941722, #2006388, and #2125511, Office of Naval Research, and Air Force Office of Scientific Research for their support.

References

[1] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. In *NIPS 2013 Deep Learning Workshop*, 2013.

[3] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. de Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*, 2017.

[4] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, and S. Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1146–1155. PMLR, 06–11 Aug 2017. URL `http://proceedings.mlr.press/v70/foerster17b.html`.

[5] M. Bowling and M. Veloso. Rational and convergent learning in stochastic games. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'01, page 1021–1026, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558608125.

[6] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artif. Intell.*, 136(2):215–250, Apr. 2002. ISSN 0004-3702. doi:10.1016/S0004-3702(02)00121-2. URL `https://doi.org/10.1016/S0004-3702(02)00121-2`.

[7] E. M. de Cote, A. Lazaric, and M. Restelli. Learning to cooperate in multi-agent social dilemmas. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, page 783–785, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933034. doi:10.1145/1160633.1160770. URL `https://doi.org/10.1145/1160633.1160770`.

[8] S. Abdallah and V. Lesser. Non-linear dynamics in multiagent reinforcement learning algorithms. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '08, page 1321–1324, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9780981738123.

[9] J. Foerster, R. Y. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, and I. Mordatch. Learning with opponent-learning awareness. AAMAS '18, page 122–130, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

[10] A. Xie, D. Losey, R. Tolsma, C. Finn, and D. Sadigh. Learning latent representations to influence multi-agent interaction. In *Proceedings of the 4th Conference on Robot Learning (CoRL)*, 11 2020.

[11] B. C. da Silva, E. W. Basso, A. L. C. Bazzan, and P. M. Engel. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 217–224, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi:10.1145/1143844.1143872. URL `https://doi.org/10.1145/1143844.1143872`.

[12] P. Hernandez-Leal, Y. Zhan, M. E. Taylor, L. E. Sucar, and E. Munoz de Cote. Detecting switches against non-stationary opponents. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, page 920–921, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems.

[13] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. D. Dragan. Planning for autonomous cars that leverage effects on human actions. In *Proceedings of Robotics: Science and Systems (RSS)*, June 2016. doi:10.15607/RSS.2016.XII.029.

[14] K. Tuyls and G. Weiss. Multiagent learning: Basics, challenges, and prospects. *AI Magazine*, 33(3):41, Sep. 2012. doi:10.1609/aimag.v33i3.2426. URL `https://ojs.aaai.org/index.php/aimagazine/article/view/2426`.

[15] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *In Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337. Morgan Kaufmann, 1993.

[16] W. Z. Wang, M. Beliaev, E. Biyik, D. A. Lazar, R. Pedarsani, and D. Sadigh. Emergent prosociality in multi-agent games through gifting. *CoRR*, abs/2105.06593, 2021. URL `https://arxiv.org/abs/2105.06593`.

[17] M. Carroll, R. Shah, M. Ho, T. Griffiths, S. Seshia, P. Abbeel, and A. Dragan. On the utility of learning about humans for human-ai coordination. *Advances in Neural Information Processing Systems*, 32, 2019. ISSN 1049-5258.

[18] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6382–6393, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[19] S. Iqbal and F. Sha. Coordinated exploration via intrinsic rewards for multi-agent reinforcement learning. *arXiv preprint arXiv:1905.12127*, 2019.

[20] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL `https://proceedings.neurips.cc/paper/2016/file/c7635bfd99248a2cdef8249ef7bfbef4-Paper.pdf`.

[21] D. P. Losey, M. Li, J. Bohg, and D. Sadigh. Learning from my partner's actions: Roles in decentralized robot teams. In *Proceedings of the 3rd Conference on Robot Learning (CoRL)*, 2019.

[22] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[23] H. He, J. Boyd-Graber, K. Kwok, and H. Daumé, III. Opponent modeling in deep reinforcement learning. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1804–1813, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `http://proceedings.mlr.press/v48/he16.html`.

[24] D. Sadigh, N. Landolfi, S. S. Sastry, S. A. Seshia, and A. D. Dragan. Planning for cars that coordinate with people: Leveraging effects on human actions for planning and active information gathering over human internal state. *Autonomous Robots (AURO)*, 42(7):1405–1426, October 2018. ISSN 1573-7527. doi:10.1007/s10514-018-9746-1.

[25] Z. Zhu, E. Biyik, and D. Sadigh. Multi-agent safe planning with gaussian processes. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2020.

[26] A. Lupu and D. Precup. Gifting in multi-agent reinforcement learning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, page 789–797, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450375184.

[27] J. Yang, A. Li, M. Farajtabar, P. Sunehag, E. Hughes, and H. Zha. Learning to incentivize other learning agents. *CoRR*, abs/2006.06051, 2020. URL `https://arxiv.org/abs/2006.06051`.

[28] K. Ndousse, D. Eck, S. Levine, and N. Jaques. Learning social learning. 2020. URL https://arxiv.org/abs/2010.00581.

[29] H. de Weerd, R. Verbrugge, and B. Verheij. How much does it help to know what she knows you know? an agent-based simulation study. *Artificial Intelligence*, 199-200:67–92, 2013. ISSN 0004-3702. doi:https://doi.org/10.1016/j.artint.2013.05.004. URL https://www.sciencedirect.com/science/article/pii/S0004370213000441.

[30] F. B. von der Osten, M. Kirley, and T. Miller. The minds of many: Opponent modeling in a stochastic game. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3845–3851, 2017. doi:10.24963/ijcai.2017/537. URL https://doi.org/10.24963/ijcai.2017/537.

[31] M. Watter, J. T. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. NIPS'15, Cambridge, MA, USA, 2015. MIT Press.

[32] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5331–5340. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/rakelly19a.html.

[33] R. Julian, E. Heiden, Z. He, H. Zhang, S. Schaal, J. Lim, G. Sukhatme, and K. Hausman. Scaling simulation-to-real transfer by learning composable robot skills. In J. Xiao, T. Kröger, and O. Khatib, editors, *Proceedings of the 2018 International Symposium on Experimental Robotics*, pages 267–279, Cham, 2020. Springer International Publishing. ISBN 978-3-030-33950-0.

[34] Y.-M. De Hauwere, P. Vrancx, and A. Nowé. Learning multi-agent state space representations. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*, AAMAS '10, page 715–722, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9780982657119.

[35] G. Tesauro. Extending q-learning to general adaptive multi-agent systems. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2004. URL https://proceedings.neurips.cc/paper/2003/file/e71e5cd119bbc5797164fb0cd7fd94a4-Paper.pdf.

[36] A. Grover, M. Al-Shedivat, J. Gupta, Y. Burda, and H. Edwards. Learning policy representations in multiagent systems. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1802–1811. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/grover18a.html.

[37] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. 2017. URL https://arxiv.org/abs/1611.01144.

[38] C. J. Maddison, A. Mnih, and Y. W. Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *International Conference on Learning Representations*, 2017.

[39] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. AAAI'08. AAAI Press, 2008. ISBN 9781577353683.

[40] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2778–2787. JMLR.org, 2017.

[41] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 1479–1487, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

[42] K. Byl and R. Tedrake. Metastable walking machines. *The International Journal of Robotics Research*, 28(8):1040–1064, 2009. doi:10.1177/0278364909340446. URL `https://doi.org/10.1177/0278364909340446`.

[43] G. Berseth, D. Geng, C. M. Devin, N. Rhinehart, C. Finn, D. Jayaraman, and S. Levine. Smirl: Surprise minimizing reinforcement learning in unstable environments. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=cPZOyoDloxl`.

[44] F. Doshi-Velez and G. D. Konidaris. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*, 2016.

[45] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018. URL `http://proceedings.mlr.press/v80/haarnoja18b.html`.

[46] R. Vidal, O. Shakernia, H. Kim, D. Shim, and S. Sastry. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. *IEEE Transactions on Robotics and Automation*, 18(5):662–669, 2002. doi:10.1109/TRA.2002.804040.

[47] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. URL `http://arxiv.org/abs/1606.01540`.

[48] E. J. Gumbel. The Maxima of the Mean Largest Value and of the Range. *The Annals of Mathematical Statistics*, 25(1):76 – 84, 1954. doi:10.1214/aoms/1177728847. URL `https://doi.org/10.1214/aoms/1177728847`.

[49] C. J. Maddison, D. Tarlow, and T. Minka. A* sampling, 2015.

[50] A. Shih, A. Sawhney, J. Kondic, S. Ermon, and D. Sadigh. On the critical role of conventions in adaptive human-ai collaboration. In *9th International Conference on Learning Representations (ICLR)*, 5 2021.

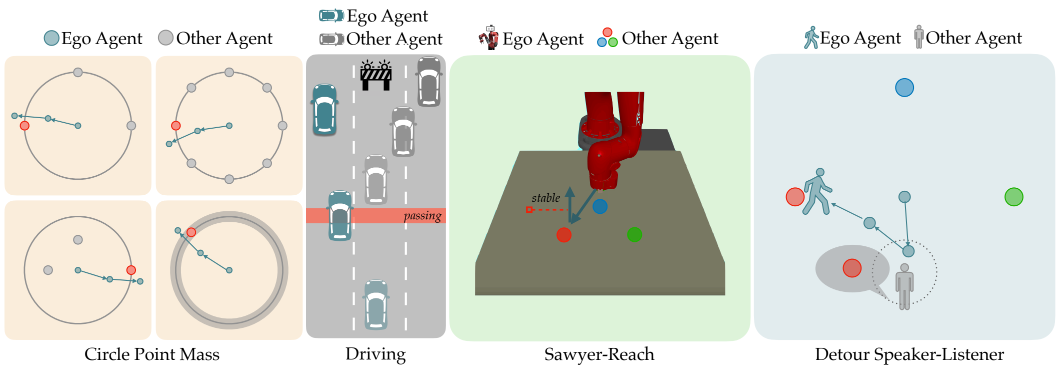

Figure 4: Stabilizing behaviors in all simulated environments. In the Circle Point Mass environments, the ego agent ends each interaction outside the circle to stabilize the opponent's strategy. In the Driving environment, the ego agent passes on the left before the red line, establishing the convention that faster vehicles pass on the left. In Sawyer-Reach, the ego agent ends the interaction above a fixed plane in the z-axis to stabilize the opponent's choice of goal location. In Detour Speaker-Listener, the ego agent takes a detour to go near the speaker to stabilize the opponent's communication strategy.

In the Appendix, we provide specific implementation details with relevant hyperparameters, and we also describe further details of our simulated environments, including a discussion about the types of opponent strategies we consider. We provide qualitative analysis of trajectories from interactions at `https://sites.google.com/view/stable-marl/`.

A Implementation Details

Both the encoder and decoder are implemented as multilayer perceptrons (MLPs) with 2 fully connected layers of hidden size 256. The encoder outputs the predicted latent strategy z^{j+1} with latent dimension 10 given tuples of (s, a, r, s') from the previous interaction τ^j. The decoder then reconstructs the next states and rewards given the states, actions, and predicted latent strategy. To train the ego agent's policy, we use soft actor-critic (SAC) [45]. Both the actor π and critic Q are MLPs with 2 fully connected layers of hidden size 256 conditioned on state s and predicted latent strategies z^j, z^{j-1}. With a modified reward function, we use the same parameter update equations as [10] to update the encoder, decoder, actor, and critic. For the combination of task and stability reward, we used an equal weighting of $\beta = 0.5$.

Discrete Latent Variables. For the discrete latent variables, we use temperature $\lambda = 1$ for the Gumbel-Softmax computation. We use the Straight-Through Gumbel-Estimator, where the latent representation z is discretized using $\arg\max$ for the forward pass but the continuous approximation is used for the backward pass. Concretely, the Gumbel-Max sampling method allows us to produce latent strategies from a categorical distribution with probabilities produced by the encoder:

$$z^j = \text{one\_hot}(\arg\max_i [g_i + \log \mathcal{E}_\phi(\tau^{j-1})_i], \tag{3}$$

where $g_1 \dots g_k \sim \text{Gumbel}(0, 1)$ [48, 49]. For the backward pass, the continuous, differentiable approximation with temperature λ is as follows:

$$\tilde{z}_i^j = \frac{e^{[g_i + \log \mathcal{E}_\phi(\tau^{j-1})_i]/\lambda}}{\sum_{a=1}^k e^{[g_a + \log \mathcal{E}_\phi(\tau^{j-1})_a]/\lambda}} \tag{4}$$

B Further Details of Simulated Environments

The simulated environments with trajectories of stabilizing behavior are shown in Fig. 4.

Circle Point Mass. In the Circle Point Mass environments, the ego agent observes its own position and does not observe the opponent's position. The ego agent has a continuous 2-dimensional action space determining the agent's velocity. The task reward is defined as the Euclidean distance between the ego and opponent position: $\mathcal{R}_{\text{task}} = -||s - z||_2$. The oracle observes the opponent's position. The interaction ends after 50 timesteps.

Driving. In the Driving environment, the ego agent tries to consistently pass the opponent. The ego agent observes both their own position and the opponent's position, but does not observe the opponent's strategy or intent of which lane they will merge towards. The ego agent has a continuous 1-dimensional action space determining the agent's lateral velocity. The task reward is defined as -1 if a collision occurs 0 if no collision occurs. The oracle observes the opponent's intended choice of lane. The interaction ends after 10 timesteps.

Sawyer-Reach. In the Sawyer-Reach environment, the ego agent must try to reach one of three opponent's positions on a table. The ego agent observes their own end-effector position and does not observe the opponent's position. The ego agent has a continuous 3-dimensional action space determining the end effectors's velocity through the use of a mocap. The reward function is defined as the Euclidean distance between the ego agent's end-effector and the opponent's position: $\mathcal{R}_{\text{task}} = -||s - z||_2$. The oracle observes the opponent's position and the interaction ends after 50 timesteps. Unlike in the Circle Point Mass, the movement of the ego agent's end-effector is limited by the simulated physical limits of the Sawyer robot.

Detour Speaker-Listener. In the Detour Speaker-Listener environment, the ego agent must try to reach one of three unknown goal locations by learning how to interpret the communication of the opponent speaker. The ego agent observes their own position as well as the communication from the opponent. The opponent observes the true underlying goal and uses their strategy to decide how to communicate the goal to the ego agent. The ego agent has a continuous 2-dimensional action space determining the agent's velocity. The reward function is defined as the Euclidean distance between the ego agent's position and the opponent's position: $\mathcal{R}_{\text{task}} = -||s - z||_2$. The oracle observes the opponent position and the interaction ends after 50 timesteps. If the ego agent has not gone near the speaker within the current interaction, the opponent's communication strategy changes with probability 0.5.

In this environment, ideal behavior would be to stabilize the opponent into maintaining the same communication method, so learning how to map the communication to the correct goal landmark becomes easy. If the ego listener agent cannot interpret the speaker's communication, the listener would likely go to the average position (centroid) of all possible opponent positions.

C Equally Beneficial Latent Strategies.

We mainly consider settings where each latent strategy is equally optimal in terms of the task reward, meaning for any fixed start state $s_1 \in \mathcal{S}$, $V^*(s_1, z)$ is constant for all $z \in \mathcal{Z}$. However, it may not be possible to stabilize to all strategies. While our approach can be applied to scenarios where some latent strategies are better than others, in many multi-agent environments, there are multiple possible equally optimal opponent strategies, and the symmetry of the strategies causes the opponent to indecisively switch between strategies in an unstable manner. Thus, stabilizing in these environments can be seen as a symmetry-breaking method to influence the opponent to stay at a stable strategy, forming multi-agent conventions [50]. We show promising results in one setting with unequally beneficial latent strategies (see row 3, Fig. 3). Future work should further consider environments where there are unequally beneficial strategies for each agent.

D Effect of the Stability Weight

The stability weight β governs the relative importance of optimizing for the stability reward versus the task reward. We investigate the effect of choosing a stability weight on SILI's performance in Fig. 5 with the Circle (3 Goals) environment. We vary β from 0 to 1, where $\beta = 0$ is an instance where the agent only optimizes for the task reward and $\beta = 1$ is where the agent only optimizes for the stability reward. From Fig. 5, we see that the agent learns how to stabilize the opponent's strategy as long as β is large enough ($\beta \geq 0.4$). However, setting β too large can result in the agent optimizing less for the task reward. There is an implicit ordering in which the ego agent should first learn to stabilize the opponent's strategy, and then consequently learn a best response to the

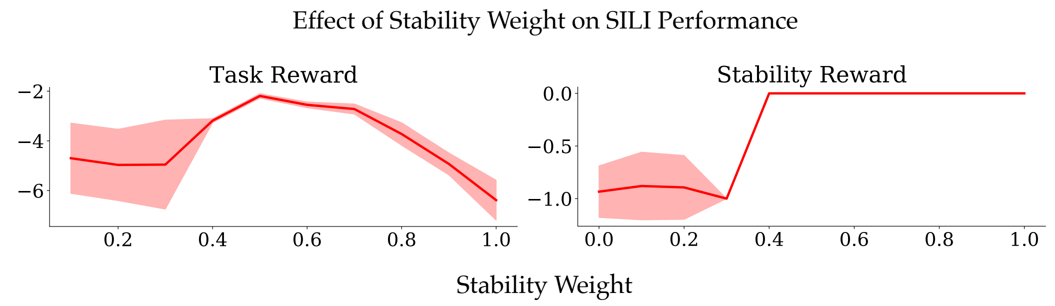

Figure 5: Effect of the stability weight on the task and stability reward in the Circle (3 Goals) environment. (Left) There appears to be a sweet spot between $0.4 \leq \beta \leq 0.8$ where SILI achieves the largest task reward. When β is too small, the agent does not learn to stabilize the opponent's strategy, so the task reward remains sub-optimal. When β is too large, the agent focuses only on stabilizing the opponent's strategy without trying to optimize for the task reward. (Right) When $\beta \geq 0.4$, the stability reward is maximized. This suggests that in practice, annealing the stability weight β may be beneficial, as the agent can first learn to stabilize the opponent's strategy, and then learn to maximize the task reward with respect to a stable opponent strategy.

fixed opponent's strategy. Thus, in practice, we found that annealing the stability weight can be an effective method to avoid needing to tune the β hyperparameter.