PSIPHI-LEARNING: REINFORCEMENT LEARNING WITH DEMONSTRATIONS USING SUCCESSOR FEATURES AND INVERSE TD LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We study reinforcement learning (RL) with no-reward demonstrations, a setting in which an RL agent has access to additional data from the interaction of other agents with the same environment. However, it has no access to the rewards or goals of these agents, and their objectives and levels of expertise may vary widely. These assumptions are common in multi-agent settings, such as autonomous driving. To effectively use this data, we turn to the framework of successor features. This allows us to disentangle shared features and dynamics of the environment from agent-specific rewards and policies. We propose a multi-task inverse reinforcement learning (IRL) algorithm, called *inverse temporal difference learning* (ITD), that learns shared state features, alongside per-agent successor features and preference vectors, purely from demonstrations without reward labels. We further show how to seamlessly integrate ITD with learning from online environment interactions, arriving at a novel algorithm for reinforcement learning with demonstrations, called $\Psi\Phi$-learning (pronounced 'Sci-Fi'). We provide empirical evidence for the effectiveness of $\Psi\Phi$-learning as a method for improving RL, IRL, imitation, and few-shot transfer, and derive worst-case bounds for its performance in zero-shot transfer to new tasks.

1 INTRODUCTION

If artificial agents are to be effective in the real world, they will need to thrive in environments populated by other agents. Agents are typically goal-directed, sometimes by definition (Franklin & Graesser, 1996). While their goals can be different, they often depend on shared salient features of the environment, and may be able to interact with and affect the environment in similar ways. Humans and other animals make ready use of these similarities to other agents while learning (Henrich, 2017; Laland, 2018). We can observe the goal-directed behaviours of other humans, and combine these observations with our own experiences, to quickly learn how to achieve our own goals. If reinforcement learning (RL, Sutton & Barto, 2018) agents could similarly interpret the behaviour of others, they could learn more efficiently, relying less on solitary trial and error.

To this end, we formalise and address a problem setting in which an agent (the 'ego-agent') is given access to observations and actions drawn from the experiences of other goal-directed agents interacting with the same environment, but pursuing distinct goals. These observed trajectories are unlabelled in the sense that they lack the goals or rewards of the other agents. This type of data is readily available in many real-world settings, either from (i) observing other agents acting simultaneously with the ego-agent in the same (multi-agent) environment, or (ii) multi-task demonstrations collected independently from the ego-agent's experiences. Consider autonomous driving as a motivating example: the car can observe the decisions of many nearby human drivers with various preferences and destinations, or may have access to a large offline dataset of such demonstrations. Because the other agents are pursuing their own varied goals, it can be difficult to directly use this information with conventional imitation learning methods (Widrow & Smith, 1964) or inverse RL (IRL) (Ng et al., 2000; Ziebart et al., 2008).

While the ego-agent should not copy other agents directly, it is likely that the behaviour of all agents depends on shared features of the environment. To disentangle such shared features from

agent-specific goals, we turn to the framework of successor features (Dayan, 1993; Kulkarni et al., 2016; Barreto et al., 2017). Successor features are a representation that captures the sum of state features an agent's policy will reach in the future. An agent's goal is represented separately as a preference vector.

In this paper, we demonstrate how a reinforcement learner can benefit from multi-task demonstrations using the framework of successor features. The key contributions are:

1. **Offline multi-task IRL:** We propose an inverse RL algorithm, called *inverse temporal difference* (ITD) *learning*. Using only demonstrations, we learn shared state features, alongside per-agent successor features and inferred preferences. The reward functions can be trivially computed from these learned quantities. We show empirically that ITD achieves superior or comparable performance to prior methods.
2. **RL with no-reward demonstrations:** By combining ITD with learning from environment interactions, we arrive at a novel algorithm for RL with unlabelled demonstrations, called $\Psi\Phi$-learning (pronounced 'Sci-Fi'). $\Psi\Phi$-learning is compatible with sub-optimal demonstrations. It treats the demonstrated trajectories as being soft-optimal under *some* task and employs ITD to recover successor features for the demonstrators' policies. $\Psi\Phi$-learning inherits the unbiased, asymptotic performance of RL methods while leveraging the provided demonstrations with ITD. When the goals of any of the demonstrators are even partially aligned with the $\Psi\Phi$-learner, this enables much faster learning than solitary RL. Otherwise, when the demonstrations are not useful or even misleading, it gracefully falls back to standard RL, unlike naïve behaviour cloning or IRL.
3. **Few-shot adaptation with task inference:** Taking full-advantage of the successor features framework, our $\Psi\Phi$-learner can even adapt zero-shot to new goals it has never seen or experienced during training, but which are partially aligned with the demonstrated goals. This is possible due to the disentanglement of representations into task-specific features (i.e., preferences) and shared state features. We can efficiently update the task-specific preferences and rely on generalised policy improvement for safe policy updates. We derive worst-case bounds for the performance of $\Psi\Phi$-learning in zero-shot transfer to new tasks.

We evaluate $\Psi\Phi$-learning in a set of grid-world environments, a traffic-flow simulator (Leurent, 2018), and a task from the ProcGen suite Cobbe et al. (2020), observing advantages over vanilla RL, imitation learning Reddy et al. (2019); Ho & Ermon (2016), and auxiliary-task baselines Hernandez-Leal et al. (2017). Thanks to the shared state features between the ITD and RL components, we find empirically that the $\Psi\Phi$-learner not only improves its ego-learning with demonstrations, but also enhances its ability to model others agents using its own experience.

2 Background and Problem Setting

We consider a world that can be represented as an infinite horizon controlled Markov process (CMP) given by the tuple: $\mathcal{C} \triangleq \langle \mathcal{S}, \mathcal{A}, P, \gamma \rangle$. $\mathcal{S}$ and $\mathcal{A}$ represent the continuous state and discrete action spaces, respectively, $\mathbf{s}' \sim P(\cdot|\mathbf{s}, \mathbf{a})$ describes the transition dynamics and γ is the discount factor. A **task** is formulated as a Markov decision process (MDP, Puterman, 2014), characterised by a reward function, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, i.e., $M \triangleq \langle \mathcal{C}, R \rangle$.

The goal of an agent is to find a policy which maps from states to a probability distribution over actions, $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, maximising the expected discounted sum of rewards $G^R \triangleq \sum_{t=0}^{\infty} \gamma^t R(\mathbf{s}_t, \mathbf{a}_t)$. The action-value function of the policy π is given by $Q^{\pi,R}(\mathbf{s}, \mathbf{a}) \triangleq \mathbb{E}^{\mathcal{C},\pi}\left[G^R \middle| \mathbf{s}_0 = \mathbf{s}, \mathbf{a}_0 = \mathbf{a}\right]$, where $\mathbb{E}^{\mathcal{C},\pi}[\cdot]$ denotes expected value when following policy π in environment $\mathcal{C}$.

2.1 RL with No-Reward Demonstrations

We are interested in settings in which, in addition to an environment $\mathcal{C}$, the agent has also access to **demonstrations without rewards**, i.e., behavioural data of mixed and unknown quality. The demonstrations are generated by other agents, whose goals and levels of expertise are unknown, and who have no incentive to educate the controlled agent. We will refer to the controlled agent, i.e., reinforcement learner, as the 'ego-agent' and to the agents that generated the demonstrations as 'other-agents'. We denote the demonstrations with $\mathcal{D} = \{\tau_1, \tau_2, \ldots, \tau_N\}$, where the trajectory

$\tau \triangleq (\mathbf{s}_0, \mathbf{a}_0, \ldots, \mathbf{s}_T, \mathbf{a}_T; k)$ is generated by the k-th agent. Note that each trajectory does include an identifier of the agent that generated it. The ego-agent also gathers its own experience by interacting with the environment, collecting data $\mathcal{B} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}', r^{\text{ego}})\}$. Due to the lack of reward annotations in $\mathcal{D}$, and the fact that the data may be irrelevant to the ego-agent's task, it is not trivial to combine demonstrations from $\mathcal{D}$ with the ego-agent's experience $\mathcal{B}$.

2.2 Successor Features and Cumulants

To make use of the demonstrations $\mathcal{D}$, we wish to capture the notion that while the agents' rewards may differ, they share the same environment. To do so we turn to the framework of successor features (SFs) (Barreto et al., 2017), in which rewards are decomposed into cumulants and preferences:

Definition 1 (Cumulants and Preferences)**.** *The (one-step) rewards are decomposed into task-agnostic* cumulants $\Phi(\mathbf{s}, \mathbf{a}) \in \mathbb{R}^d$*, and task-specific* preferences $\mathbf{w} \in \mathbb{R}^d$*:*

$$R^{\mathbf{w}}(\mathbf{s}, \mathbf{a}) \triangleq \Phi(\mathbf{s}, \mathbf{a})^\top \mathbf{w}\,. \tag{1}$$

The preferences $\mathbf{w}$ are a representation of a possible goal in the world $\mathcal{C}$, in the sense that each $\mathbf{w}$ gives rise to a task $M^{\mathbf{w}} = \langle \mathcal{C}, R^{\mathbf{w}} \rangle$. We use 'task', 'goal', and 'preferences' interchangeably when context makes it clear whether we are referring to $\mathbf{w}$ itself, or the corresponding $M^{\mathbf{w}}$ or $R^{\mathbf{w}}$. The action-value function for a policy π in $M^{\mathbf{w}}$ is then a function of the preferences $\mathbf{w}$ and the π's successor features.

Definition 2 (Successor Features)**.** *For a given discount factor $\gamma \in [0, 1)$, policy π and cumulants $\Phi(\mathbf{s}, \mathbf{a}) \in \mathbb{R}^d$, the successor features (SFs) for a state $\mathbf{s}$ and action $\mathbf{a}$ are:*

$$\Psi^\pi(\mathbf{s}, \mathbf{a}) \triangleq \mathbb{E}^{\mathcal{C}, \pi}\left[\sum_{t=0}^{\infty} \gamma^t \Phi(\mathbf{s}_t, \mathbf{a}_t) \middle| \mathbf{s}_0 = \mathbf{s}, \mathbf{a}_0 = \mathbf{a}\right]. \tag{2}$$

The i-th component of $\Psi^\pi(\mathbf{s}, \mathbf{a})$ gives the expected discounted sum of $\Phi(\mathbf{s}, \mathbf{a})$'s i-th component, when starting from state $\mathbf{s}$, taking action $\mathbf{a}$ and then following policy π. Intuitively, cumulants Φ can be seen as a vector-valued reward function and SFs Ψ^π the corresponding vector-valued state-action value function for policy π.

An action-value function is then given by the dot product of the preferences $\mathbf{w}$ and π's SFs:

$$Q^{\pi, \mathbf{w}}(\mathbf{s}, \mathbf{a}) = \Psi^\pi(\mathbf{s}, \mathbf{a})^\top \mathbf{w}\,. \tag{3}$$

Proof. See Appendix D. □

Note that if we have Ψ^π, the value of π for a new preference $\mathbf{w}'$ can be easily computed. This property allows the successor features of a set of policies to be repurposed for accelerating policy updates, as follows.

Definition 3 (Generalised Policy Improvement)**.** *Given a set of policies $\Pi = \{\pi_1, \ldots, \pi_K\}$ and a task with reward function R, generalised policy improvement (GPI) is the definition of a policy π' such that*

$$Q^{\pi', R}(\mathbf{s}, \mathbf{a}) \geq \sup_{\pi \in \Pi} Q^{\pi, R}(\mathbf{s}, \mathbf{a})\,, \forall \mathbf{s} \in \mathcal{S}, \mathbf{a} \in \mathcal{A}\,. \tag{4}$$

Provided the SFs of a set of policies, i.e., $\{\Psi^{\pi^k}\}_{k=1}^K$, we can apply GPI to derive a new policy π' whose performance on a task $\mathbf{w}$ is no worse that the performance of any of $\pi \in \Pi$ on the same task, given by

$$\pi'(\mathbf{s}) = \arg\max_a \max_{\pi \in \Pi} \Psi^\pi(\mathbf{s}, a)^\top \mathbf{w}\,. \tag{5}$$

Eqn. (5) suggests that if we could estimate the SFs of other agents, we could utilise them for improving the ego-agent's policy with GPI. However, to do so with conventional methods we would require access to their rewards, cumulants and/or preferences. In our setting, we can only observe their sequence of states and actions (Section 2.1). Next, we introduce our method that only requires no-reward demonstrations to estimate SFs and can be integrated seamlessly with GPI for accelerating reinforcement learning.

3 ACCELERATING RL WITH DEMONSTRATIONS

We now present a novel method, $\Psi\Phi$-learning , that leverages reward-free demonstrations to accelerate RL.Our approach consists of two components: (i) a novel inverse reinforcement learning algorithm, called *inverse temporal difference* (ITD) *learning*, for learning cumulants, per-agent successor features and corresponding agent preferences from demonstration without reward labels, and (ii) a novel RL algorithm that combines ITD with generalised policy improvement (GPI, Section 2.2).

3.1 INVERSE TEMPORAL DIFFERENCE LEARNING

Given demonstrations without rewards, $\mathcal{D}$, we model the agents that generated the data as soft-optimal for an *unknown* task. In particular, the k-th agent's policy is soft-optimal under task $\mathbf{w}^k$ and is given by

$$\pi^k(\mathbf{a}|\mathbf{s}) = \frac{\exp(\Psi^{\pi^k}(\mathbf{s},\mathbf{a})^\top \mathbf{w}^k)}{\sum_a \exp(\Psi^{\pi^k}(\mathbf{s},a)^\top \mathbf{w}^k)}\,, \forall \mathbf{s}\in\mathcal{S}, \mathbf{a}\in\mathcal{A}\,. \tag{6}$$

We choose to represent the action-value functions of the other agents with their SFs and preferences to enable GPI, and to expose task- and policy-agnostic structure in the form of shared cumulants Φ. The k-th agent's successor features are temporally consistent with these cumulants Φ

$$\Psi^{\pi^k}(\mathbf{s},\mathbf{a}) = \Phi(\mathbf{s},\mathbf{a}) + \gamma \mathbb{E}^{\mathcal{C},\pi^k}\left[\Psi^{\pi^k}(\mathbf{s}',\pi^k(\mathbf{s}))\right]\,. \tag{7}$$

To learn these quantities from K demonstrators, we parameterise the SFs with $\boldsymbol{\theta}^{\Psi^k}$, preferences with $\mathbf{w}^k$, and shared cumulants with $\boldsymbol{\theta}_\Phi$. A schematic of the model architecture and further details are provided in Appendix C. The parameters are learned by minimising a behavioural cloning and SFs TD loss based on equations (6) and (7).

Behavioural cloning loss. Given demonstrations generated only by the k-th agent, i.e., $\mathcal{D}^k \subset \mathcal{D}$, we train its successor features $\boldsymbol{\theta}_{\Psi^k}$ and the preferences $\mathbf{w}^k$ by minimising the negative log-likelihood of the demonstrations

$$\mathcal{L}_{\text{BC-}Q}(\boldsymbol{\theta}_{\Psi^k}, \mathbf{w}^k) \triangleq -\mathop{\mathbb{E}}_{\tau\sim\mathcal{D}^k} \log \frac{\exp(\Psi(\mathbf{s}_t,\mathbf{a}_t;\boldsymbol{\theta}_{\Psi^k})^\top \mathbf{w}^k)}{\sum_a \exp(\Psi(\mathbf{s}_t,a;\boldsymbol{\theta}_{\Psi^k})^\top \mathbf{w}^k)}\,. \tag{8}$$

Importantly, Eqn. (8) reflects the fact that along a trajectory τ, the successor features are a function of the state and action at each time-step, $\mathbf{s}_t$ and $\mathbf{a}_t$, while the preferences $\mathbf{w}^k$ are learnable but consistent across time and trajectories. A sparsity prior, i.e., $\mathcal{L}_1$ loss, on preferences $\mathbf{w}^k$ is also used to promote disentangled cumulant dimensions, see Figure 8.

Inverse temporal difference loss. The cumulant parameters $\boldsymbol{\theta}_\Phi$ are trained to be TD-consistent with all agents' successor features. This procedure inverts[1] the standard TD-learning framework for SFs (Dayan, 1993; Barreto et al., 2017) where they are trained to be consistent with a fixed cumulant Φ. Instead, we first train the SFs and preference vectors to 'explain' the other agents' behaviour with the behavioural cloning loss Eqn. (8), and then train $\boldsymbol{\theta}_\Phi$ to be consistent with these successor features by minimising

$$\mathcal{L}_{\text{ITD}}(\boldsymbol{\theta}_\Phi) \triangleq \mathop{\mathbb{E}}_{(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1},\mathbf{a}_{t+1},k)\sim\mathcal{D}} \|\Phi(\mathbf{s}_t,\mathbf{a}_t;\boldsymbol{\theta}_\Phi) + \underbrace{\gamma\Psi(\mathbf{s}_{t+1},\mathbf{a}_{t+1};\tilde{\boldsymbol{\theta}}_{\Psi^k}) - \Psi(\mathbf{s}_t,\mathbf{a}_t;\boldsymbol{\theta}_{\Psi^k})}_{\texttt{stop\_gradient}}\|\,. \tag{9}$$

In practice, our ITD-learning algorithm alternates between minimising $\mathcal{L}_{\text{BC-}Q}$ for training the successor features and preferences, and $\mathcal{L}_{\text{ITD}}$ for training the shared cumulants, provided only with no-reward demonstrations.From the definition of cumulants and preferences, we can recover the k-th demonstrator's reward, by applying Eqn. (1), i.e., $R^k(\mathbf{s},\mathbf{a}) \approx \Phi(\mathbf{s},\mathbf{a};\boldsymbol{\theta}_\Phi)^\top \mathbf{w}^k$. Our ITD algorithm returns both Q-functions that can be used for imitating a demonstrator and an explicit reward function for each agent, requiring only access to demonstrations without any online interaction with the simulator. Hence ITD-learning is an offline multi-task IRL algorithm. ITD is summarised in Algorithm 1.

Single-task setting. To gain more intuition about the ITD algorithm, consider the simpler case of performing IRL with demonstrations from a single policy. This obviates the need for a representation

[1] Hence the name *inverse TD learning*.

of preferences, so we can use $\mathbf{w} = 1$. In this case Ψ is the action-value function Q and Φ is simply the reward R. Minimising (8) reduces to finding a Q-function whose softmax gives the observed policy, and minimising (9) finds a scalar reward that explains the Q-function. Our more general formulation, with cumulants Φ in place of a scalar reward, allows us to perform ITD-learning on demonstrations from many policies, and to efficiently transfer to new tasks, as we show next.

3.2 $\Psi\Phi$-LEARNING WITH NO-REWARD DEMONSTRATIONS

Now we present our main contribution, $\Psi\Phi$-learning , which combines our ITD inverse RL algorithm with RL and GPI, using no-reward demonstrations from other agents to accelerating the ego-agent's learning. $\Psi\Phi$-learning is summarised in Algorithm 1.

$\Psi\Phi$-learning is an off-policy algorithm based on Q-learning (Watkins & Dayan, 1992; Mnih et al., 2013). The action-value function is represented with successor features, Ψ^{ego}, and preferences, $\mathbf{w}^{\text{ego}}$, as in Eqn. (3). The ego-agent interacts with the environment, storing its experience in a replay buffer, $\mathcal{B} \leftarrow \mathcal{B} \cup \{(\mathbf{s}, \mathbf{a}, \mathbf{s}', r^{\text{ego}})\}$. The $\Psi\Phi$-learner also has estimates for the cumulants, per-agent SFs and preferences obtained with ITD from the demonstrations $\mathcal{D}$.

Reward loss. The ego-rewards, r^{ego}, are used to ground the cumulants Φ and the preferences $\mathbf{w}^{\text{ego}}$, via the loss

$$\mathcal{L}_{\text{R}}(\boldsymbol{\theta}_\Phi, \mathbf{w}^{\text{ego}}) \triangleq \mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a},r^{\text{ego}})\sim\mathcal{B}} \| \Phi(\mathbf{s}, \mathbf{a}; \boldsymbol{\theta}_\Phi)^\top \mathbf{w}^{\text{ego}} - r^{\text{ego}} \| \,. \tag{10}$$

Importantly, we share the *same* cumulants between the ITD-learning from other agents and the ego-learning, so that they span the joint space of reward functions. This can be also seen as a representation learning method, where by enforcing all agents, including the ego-agent, to share the same Φ, we transfer information about salient features of the environment from learning about one agent to benefit learning about all agents.

Temporal difference learning. The ego-agent's successor features are learned using two losses. First, we train the SFs to fit the Q-values using the Bellman error

$$\mathcal{L}_Q(\boldsymbol{\theta}_{\Psi^{\text{ego}}}) \triangleq \mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a},\mathbf{s}',r^{\text{ego}})\sim\mathcal{B}} \| \Psi(\mathbf{s}, \mathbf{a}; \boldsymbol{\theta}_\Psi)^\top \mathbf{w}^{\text{ego}} - r^{\text{ego}} - \underbrace{\gamma \max_{a'} \Psi(\mathbf{s}', \mathbf{a}'; \tilde{\boldsymbol{\theta}}_\Psi)^\top \mathbf{w}^{\text{ego}}}_{\texttt{stop-gradient}} \| \,. \tag{11}$$

We additionally train the successor features to be self-consistent (i.e. to satisfy Equation 2) using a TD loss $\mathcal{L}_{\text{TD-}\Psi}$.

$$\mathcal{L}_{\text{TD-}\Psi}(\boldsymbol{\theta}_{\Psi^{\text{ego}}}) \triangleq \mathop{\mathbb{E}}_{(\mathbf{s},\mathbf{a},\mathbf{s}',\mathbf{a}')\sim\mathcal{B}} \| \Psi(\mathbf{s}, \mathbf{a}; \boldsymbol{\theta}_\Psi) - \Phi(\mathbf{s}, \mathbf{a}; \tilde{\boldsymbol{\theta}}_\Phi) - \gamma \underbrace{\Psi(\mathbf{s}', \mathbf{a}'; \boldsymbol{\theta}_\Psi)}_{\texttt{stop-gradient}} \| \,. \tag{12}$$

GPI behavioural policy. Provided SFs estimates for the other agents, $\{\boldsymbol{\theta}_{\Psi^k}\}_{k=1}^K$, and the ego-agent $\boldsymbol{\theta}_{\Psi^{\text{ego}}}$, and inferred ego preferences, $\mathbf{w}^{\text{ego}}$, we adopt an action selection mechanism according to the GPI rule in Eqn. (5)

$$\pi^{\text{ego}}(\mathbf{s}) = \arg\max_a \max_{\boldsymbol{\theta}_\Psi} \Psi(\mathbf{s}, a; \boldsymbol{\theta}_\Psi)^\top \mathbf{w}^{\text{ego}} \,. \tag{13}$$

The GPI step lets the agent estimate the value of the demonstration policies on its current task, and then copy the policy that it predicts will be most useful. We combat model overestimation by acting pessimistically with regard to an ensemble of two successor features approximators. If the agent's estimated values are accurate and the demonstration policies are useful for the ego task, the GPI policy can obtain good performance faster than policy iteration with only the ego value function. The next section quantifies this claim.

Performance Bound. Given a set of demonstration task vectors $\{w_k\}_{k=1}^K$ and successor features for the corresponding optimal policies $\{\Psi^{\pi^k}\}_{k=1}^K$, Barreto et al. (2017) show that it is possible to bound the performance of the GPI policy on the ego-agent task w' as a function of the distance of w' from the closest demonstration task w_j, and the value approximation error of the predicted value functions $\widetilde{Q}^{\pi_i} = \Psi^{\pi_i} w'$. We extend this result to explicitly account for the reward approximation error δ_r obtained by the the learned cumulants and the SF approximation error δ_Ψ.

Theorem 1. *(Informal statement.) Let π^* be the optimal policy for the ego task w' and let π be the GPI policy obtained from $\{\tilde{Q}^{\pi_i}\}$, with δ_r, δ_Ψ the reward and successor feature approximation errors. Then for all s, a*

$$Q^*(s,a) - Q^{\pi}(s,a) \leq \frac{2}{1-\gamma}\left[\phi_{\max} \min_j \|w' - w_j\| + 2\delta_r + \|w'\|\delta_\Psi + \frac{\delta_r}{1-\gamma}\right]. \tag{14}$$

Proof. See Appendix D for a formal statement. □

In settings where the agent has a good reward and SFs approximation, and the ego task vector w' is close to the demonstration tasks, Theorem 1 says that the ego-agent will attain near-optimal performance from the start of training.

4 Experiments

We conduct a series of experiments to determine how well $\Psi\Phi$-learning functions as an RL, IRL, imitation learning, and transfer learning algorithm.

Baselines. We benchmark against the following methods: (i) **DQN:** Deep Q-learning Mnih et al. (2013), (ii) **BC:** Behaviour Cloning, a simple imitation learning method in which we learn $p(\mathbf{a}|\mathbf{s})$ via supervised learning on the demonstration data, (iii) **DQN+BC-AUX:** DQN with an additional behavior-cloning auxiliary loss Hernandez-Leal et al. (2019), (iv) **GAIL:** Generative Adversarial Imitation Learning Ho & Ermon (2016), which uses a GAN-like approach to approximate the expert policy, and (v) **SQILv2**: Soft Q Imitation Learning Reddy et al. (2019), a recently proposed imitation technique that combines imitation and RL, and works in the absence of rewards. For high-dimensional environments, we replace DQN with **PPO**, Proximal Policy Optimization Schulman et al. (2017). Both DQN and PPO are trained to optimize environment reward through experience, and do not have access to other agents' experiences.

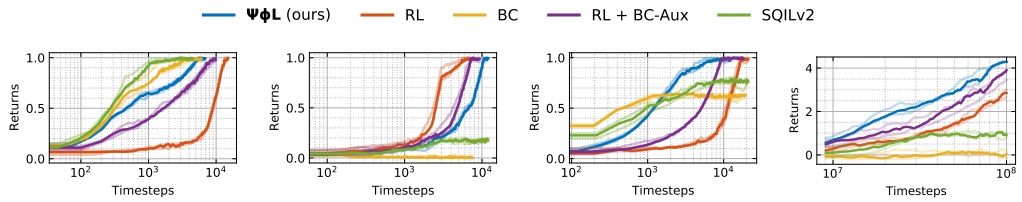
4.1 Environments

Experiments are conducted using four environments, shown in Figure **??**. We cover a broad range of problem setting, including both multi-agent and single-agent environments, as well as learning online during RL training, or offline from previously collected demonstrations.

Highway (Leurent, 2018) is a multi-agent autonomous driving environment in which the ego-agent must safely navigate around other cars and reach its goal. The other agents follow near-optimal scripted policies for various goals, depending on the scenario. In the **single-task** scenario, other agents have the same objective as the ego-agent, so their experience is directly relevant. In the **adversarial task**, the other agents do not move, and the ego-agent has to accelerate and go to a particular lane while avoiding other vehicles. Finally, in the **multi-task** scenario, the other agents and ego-agent have different preferences over target speed, preferred lane, and following distance. We consider the multi-task scenario to be the most realistic and representative of real highway driving with human drivers. In addition to highway driving, we also study the more complex **Roundabout** task. Roundabout is inherently multi-task, in that other agents randomly exit either the first or second exit, while the ego-agent must learn to take the third exit.

CoinGrid is a single-agent grid-world, environment containing goals of different colours. We collect offline trajectories of pre-trained agents with preferences for different goals. During training, the ego-agent is only rewarded for collecting a subset of the possible goals. We can then test how well the ego-agent is able to transfer to a goal that was never experienced during training (Section 4.5). This environment also enables learning easily interpretable preference vectors, allowing us to visualize how well our method works as an IRL method for inferring rewards (Section 4.3).

FruitBot is a high-dimensional, procedurally generated, single-agent environment from the OpenAI ProcGen (Cobbe et al., 2020) suite. We use FruitBot to test whether $\Psi\Phi$-learning can scale up to more complex RL environments, requiring larger deep neural network architectures that learn directly from pixels. The agent must navigate around randomly generated obstacles while collecting fruit, and avoiding other objects and walls. To create a multi-task version of FruitBot, we define additional tasks which vary agents' preferences over collecting objects in the environment, and train PPO baselines on these task variants. The ego-agent observes the states and actions of these trained agents playing the game in parallel with its own interactions.



(a) Highway: Single-task (b) Highway: Adversarial (c) Highway: Multi-task (d) FruitBot

Figure 1: Learning curves for ΨΦ-learning and baselines in three tasks in the multi-agent Highway environment (a-c), and in single-agent FruitBot (d). Tasks (a) and (b) represent extreme cases where either RL or imitation learning is irrelevant. In (a) other agents have the same task as the ego-agent, so imitation learning excels. In the adversarial task (b), other agents exhibit degenerative behaviour, so imitation learning performs extremely poorly and traditional RL (DQN) excels. In both of these extreme cases, ΨΦ-learning achieves good performance, showing it can flexibly reap the benefits of either imitation or RL as appropriate. Task (c) is most realistic; here, other agents have varied preferences and goals that may or may not relate to the ego-agent's task. ΨΦ-learning clearly outperforms baseline techniques. Similar results are shown in Fruitbot (d), showing that ΨΦ-learning scales well to high-dimensional environments, consistently outperforming baselines like PPO and SQIL. We plot mean performance over 3 runs and individual runs with alpha.

4.2 Accelerating RL with No-Reward Demonstrations

This section addresses two hypotheses: **H1:** When the unlabelled demonstrations are relevant, ΨΦ-learning can accelerate or improve performance of the ego-agent when learning with online RL; and **H2:** If the demonstrations are irrelevant, biased, or are generated by sub-optimal demonstrators, ΨΦ-learning can perform at least as well as standard RL.

Figure 1 shows the results of ΨΦ-learning and the baselines in the Highway and FruitBot environments. In the single-task scenario (1a), when other agents' experience is entirely relevant to the ego-agent's task, imitation learning methods like BC and SQILv2 learn fastest. DQN learns slowly because it does not use the other agents' experience. However, ΨΦ-learning achieves competitive results, out-performing DQN+BC-Aux Hernandez-Leal et al. (2019); Ndousse et al. (2020). In the adversarial task (1b), the other agents' behaviors are irrelevant for the ego-agent's task, so imitation learning (BC and SQILv2) performs poorly, while traditional RL techniques (DQN and DQN+BC-Aux) perform best. The performance of ΨΦ-learning does not suffer like other imitation learning methods; instead, it retains the performance of standard RL (**H2**). ΨΦ-learning can flexibly reap the benefits of either imitation learning or RL, depending on what is most beneficial for the task.

The multi-task scenario (1c) is the most realistic autonomous driving task, in which other agents navigate the highway with varying driving styles. Here, ΨΦ-learning clearly out-performs all other methods, suggesting it can leverage information about other agents' preferences in order to learn the underlying task structure of the environment, acclerating performance on the ego-agent's RL task (**H1**). FruitBot (1d) gives consistent results, showing that ΨΦ-learning scales well to high-dimensional, single-agent tasks while still outperforming BC, SQIL, PPO, and PPO+BC-Aux.

4.3 Inverse Reinforcement Learning

We now test hypothesis **H3:** ITD is an effective IRL method, and can accurately infer other agents' rewards. We present a quantitative and qualitative study of the rewards for other agents that are inferred by ITD, as well as the learned cumulants and preferences. Here, we focus solely on offline IRL and use only ITD to learn from offline reward-free demonstrations, without any ego-agent experience.

To quantitatively evaluate how well ITD can infer rewards, we train an RL agent on the inferred reward function, and compare the performance to other imitation learning and IRL methods. Table 3 gives the performance in terms of normalised returns on all three environments. Using ITD to infer rewards results in significantly higher performance than BC and SQIL, in two environments, and competitive performance in FruitBot. We note that unlike ΨΦ-learning , BC and SQIL directly learn

Table 1: We evaluate how well ΨΦ-learning is able to transfer to new tasks in a few-shot fashion. We construct a multi-task variant of the CoinGrid environment: The ego-agent is provided demonstrations for either capturing only red coins R or only green coins G. Then it is evaluated on 4 different tasks: collecting (i) both red and green coins R+G, (ii) collecting red and avoiding green coins R−G, (iii) avoiding red and collecting green coins −R+G and (iv) avoiding both red and green coins −R−G. A "♢" indicates methods that use a single model for all tasks, while "♣" indicates methods that require one model per task, i.e., they comprise of 4 models. Because it disentangles preferences from task representation, ΨΦ-learning is able to adapt to reach optimal performance on the new tasks after a single episode or improve intra-episode from the first episode after experiencing the first rewards. In contrast, SQIL takes 100 episodes to adapt.

	0-shot				1-shot				100-shot			
Methods	R+G	R−G	−R+G	−R−G	R+G	R−G	−R+G	−R−G	R+G	R−G	−R+G	−R−G
SQILv2$^{\clubsuit}$ (Reddy et al., 2019)	1.0±0.0	0.0±0.0	0.0±0.0	−1.0±0.0	**1.0**±0.0	0.0±0.0	0.0±0.0	−1.0±0.0	**1.0**±0.0	**1.0**±0.0	**1.0**±0.0	**1.0**±0.0
ΨΦ-learning $^{\diamondsuit}$ (ours, cf. Section 3.2)	1.0±0.0	**0.2**±0.1	**0.2**±0.1	**−0.4**±0.2	**1.0**±0.0	**1.0**±0.0	**1.0**±0.0	**1.0**±0.0	**1.0**±0.0	**1.0**±0.0	**1.0**±0.0	**1.0**±0.0

a policy from demonstrations, and do not actually infer an explicit reward function. In contrast, GAIL does infer an explicit reward function, and ITD gives consistently higher performance than GAIL in all three environments. These results demonstrate that ITD is an effective IRL technique (**H3**).

Qualitatively, we can evaluate how well the cumulants inferred by ITD in the CoinGrid environment span the space of possible goals. We compute the learned cumulants $\hat{\phi}(s)$ for each square s in the grid. Figure 8a shows the original CoinGrid game, and Figure 8b-8d shows the first three dimensions of the learned cumulant vector, $\hat{\phi}_1$-$\hat{\phi}_3$ (the rest are given in the Appendix). We find that $\hat{\phi}_1$ is most active for red coins, $\hat{\phi}_2$ for green, and $\hat{\phi}_3$ for yellow. Clearly, ITD has learned cumulant features that span the space of goals for this game. See Appendix F for more details and visualisations of the learned rewards and preferences.

4.4 Imitation Learning

Here, we investigate hypothesis **H4:** ΨΦ-learning works as an effective imitation learning method, allowing for accurate prediction of other agents' actions. To test this hypothesis, we train other agents in the Roundabout environment, then split no-reward demonstrations from these agents into a train dataset (80%) and a held-out test dataset. We use the train dataset to run ITD, which means that we use the data to learn both Φ and the Ψ and $\mathbf{w}$ for other agents. Because it is specifically designed to accurately predict other agents' actions, we use BC as the baseline. We compare this to using only ITD, and using the full ΨΦ-learning algorithm including ITD *and* learning from RL and experience to update the shared cumulants Φ.

Accuracy in predicting other agents' actions on the held-out test set is used to measure imitation learning performance. Figure 11 shows accuracy over the course of training. At each phase change marked in the figure, the ego-agent is given a new task, to test how the representation learning benefits from diverse ego-experience. We see that although BC obtains accurate train performance, it generalises poorly to the test set, reaching little over 80% accuracy. Without RL, ΨΦ-learning achieves similar performance. However, when using RL to improve imitation, ΨΦ-learning performs well on both the train and test set, achieving markedly higher accuracy ($\sim$95%) in predicting other agents' behaviour. This suggests that when ΨΦ-learning uses RL and interaction with the world to improve the estimation of the shared cumulants Φ, this in turn improves its ability to model the Q function of other agents and predict their behaviour. Further, ΨΦ-learning adapts well when the agent's goal changes, since it uses SFs to disentangle the representation of an agent's goal from environment dynamics. Taken together, these results demonstrate that ΨΦ-learning also works as a competitive imitation learning method (**H4**).

4.5 Transfer and Few-Shot Generalisation

Since SFs have been shown to improve generalization and transfer in RL, here we test hypothesis **H5:** ΨΦ-learning will be able to generalize effectively to new tasks in a few-shot transfer setting. Using the CoinGrid environment, it is possible to precisely test whether the ego-agent can generalise to a task it has never experienced during training. Specifically, we would like to determine whether: (i)

the ego-agent can generalise to tasks it was never rewarded for during training (but which it may have seen other agents demonstrate), and (ii) the ego-agent can generalise to tasks not experienced by *any* agents during training.

Table 1 shows the results of transfer experiments in which agents are given 0, 1, or 100 additional training episodes to adapt to a new task. Unlike SQIL, $\Psi\Phi$-learning is able to adapt 0-shot to obtain some reward on the new tasks, and fully adapt after a single episode to achieve the maximum reward on all transfer tasks. This is because $\Psi\Phi$-learning uses SFs to disentangle preferences (goals) in the task representation, and learn about the space of possible preferences from observing other agents. To adapt to a new task, it need only infer the correct preference vector. Task inference is trivially implemented as a least squares regression problem, see Eqn. (10): Having experienced $\mathcal{B}^{\text{new}}$ in the new task, the $\Psi\Phi$-learner identifies the preference vector for the new task by solving $\min_{\mathbf{w}} \sum_{\mathcal{B}^{\text{new}}} \mathcal{L}_{\text{R}}(\boldsymbol{\theta}_{\Phi}, \mathbf{w})$. In contrast, SQIL requires 100 episodes to reach the same performance.

5 Discussion

We have presented two major algorithmic contributions. The first, ITD, is a novel and flexible offline IRL algorithm that discovers salient task-agnostic environment features in the form of cumulants, as well as learning successor features and preference vectors for each agent which provides demonstrations. The second, $\Psi\Phi$-learning , combines ITD with RL from online experience. This makes efficient use of unlabelled demonstrations to accelerate RL, and comes with theoretical worst-case performance guarantees. We showed empirically the advantages of these algorithms over various baselines: how imitation with ITD can improve RL and enable zero-shot transfer to new tasks, and how experience from online RL can help to improve imitation in turn.

Future Work. We want to explore ways to: (i) adapt $\Psi\phi$-learning to multi-agent *strategic* settings, where coordination and opponent modelling (Albrecht & Stone, 2018) are essential and (ii) use a universal successor features approximator (Borsa et al., 2018) for ITD, overcoming its current, linear scaling with the number of distinct demonstrators.

References

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

Stefano V Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

Christopher G Atkeson and Stefan Schaal. Robot learning from demonstration. In *ICML*, volume 97, pp. 12–20. Citeseer, 1997.

Yusuf Aytar, Tobias Pfaff, David Budden, Tom Le Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube. *arXiv preprint arXiv:1805.11592*, 2018.

Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Claudio Fantacci, Jonathan Godwin, Chris Jones, Tom Hennigan, Matteo Hessel, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Lena Martens, Vladimir Mikulik, Tamara Norman, John Quan, George Papamakarios, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Wojciech Stokowiec, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL http://github.com/deepmind.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pp. 4055–4065, 2017.

André Barreto, Diana Borsa, Shaobo Hou, Gheorghe Comanici, Eser Aygün, Philippe Hamel, Daniel Toyama, Shibl Mourad, David Silver, Doina Precup, et al. The option keyboard: Combining

skills in reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 13052–13062, 2019.

André Barreto, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117 (48):30079–30087, 2020.

Lukas Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Survey: Robot programming by demonstration. *Handbook of robotics*, 59(BOOK_CHAP), 2008.

Diana Borsa, Bilal Piot, Rémi Munos, and Olivier Pietquin. Observational learning by reinforcement learning. *arXiv preprint arXiv:1706.06617*, 2017.

Diana Borsa, André Barreto, John Quan, Daniel Mankowitz, Rémi Munos, Hado van Hasselt, David Silver, and Tom Schaul. Universal successor features approximators. *arXiv preprint arXiv:1812.07626*, 2018.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Daniel S Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. *arXiv preprint arXiv:1904.06387*, 2019.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: First steps towards grounded language learning with a human in the loop. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJeXCo0cYX.

JD Choi and Kee-Eung Kim. Inverse reinforcement learning in partially observable environments. *Journal of Machine Learning Research*, 12:691–730, 2011.

Sungjoon Choi, Kyungjae Lee, and Songhwai Oh. Robust learning from demonstrations with mixed qualities using leveraged gaussian processes. *IEEE Transactions on Robotics*, 35(3):564–576, 2019.

Adam Coates, Pieter Abbeel, and Andrew Y Ng. Learning for control from multiple demonstrations. In *Proceedings of the 25th international conference on Machine learning*, pp. 144–151, 2008.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pp. 2048–2056. PMLR, 2020.

Felipe Codevilla, Matthias Miiller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–9. IEEE, 2018.

Aaron Davidson. Using artificial neural networks to model opponents in texas hold'em. *Unpublished manuscript*, 1999.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

Marc Peter Deisenroth, Peter Englert, Jan Peters, and Dieter Fox. Multi-task policy search for robotics. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3876–3881. IEEE, 2014.

Christos Dimitrakakis and Constantin A Rothkopf. Bayesian multitask inverse reinforcement learning. In *European workshop on reinforcement learning*, pp. 273–284. Springer, 2011.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pp. 1407–1416. PMLR, 2018.

Angelos Filos, Panagiotis Tigkas, Rowan McAllister, Nicholas Rhinehart, Sergey Levine, and Yarin Gal. Can autonomous vehicles identify, recover from, and adapt to distribution shifts? In *International Conference on Machine Learning*, pp. 3145–3153. PMLR, 2020.

Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016a.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pp. 49–58, 2016b.

Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *International Workshop on Agent Theories, Architectures, and Languages*, pp. 21–35. Springer, 1996.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Justin Fu, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. *arXiv preprint arXiv:1902.07742*, 2019.

Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

Daniel H Grollman and Aude Billard. Donut as i do: Learning from failed demonstrations. In *2011 IEEE International Conference on Robotics and Automation*, pp. 3804–3809. IEEE, 2011.

Steven Hansen, Will Dabney, Andre Barreto, Tom Van de Wiele, David Warde-Farley, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. *arXiv preprint arXiv:1906.05030*, 2019.

He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1804–1813, 2016.

Tom Hennigan, Trevor Cai, Tamara Norman, and Igor Babuschkin. Haiku: Sonnet for JAX, 2020. URL http://github.com/deepmind/dm-haiku.

Joseph Henrich. *The secret of our success: How culture is driving human evolution, domesticating our species, and making us smarter*. Princeton University Press, 2017.

Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*, 2017.

Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. Agent modeling as auxiliary task for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 15, pp. 31–37, 2019.

TM Heskes. Solving a huge number of simular tasks: a combination of multi-task learning and a hierarchical bayesian approach. 1998.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

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

Matt Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Feryal Behbahani, Tamara Norman, Abbas Abdolmaleki, Albin Cassirer, Fan Yang, Kate Baumli, et al. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020.

John D Hunter. Matplotlib: A 2d graphics environment. *IEEE Annals of the History of Computing*, 9 (03):90–95, 2007.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

David Janz, Jiri Hron, Przemysław Mazur, Katja Hofmann, José Miguel Hernández-Lobato, and Sebastian Tschiatschek. Successor uncertainties: exploration and uncertainty in temporal difference learning. In *Advances in Neural Information Processing Systems*, pp. 4507–4516, 2019.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International Conference on Machine Learning*, pp. 3040–3049. PMLR, 2019.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pp. 651–673. PMLR, 2018.

Arne Kesting, Martin Treiber, and Dirk Helbing. Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1928):4585–4605, 2010.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*, 2016.

Kevin N Laland. *Darwin's unfinished symphony: How culture made the human mind*. Princeton University Press, 2018.

Donghun Lee, Srivatsan Srinivasan, and Finale Doshi-Velez. Truly batch apprenticeship learning with deep successor features. *arXiv preprint arXiv:1903.10077*, 2019.

Edouard Leurent. An environment for autonomous driving decision-making. https://github.com/eleurent/highway-env, 2018.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1118–1125. IEEE, 2018.

Alan J Lockett, Charles L Chen, and Risto Miikkulainen. Evolving explicit opponent models in game playing. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pp. 2106–2113, 2007.

Marlos C Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. *arXiv preprint arXiv:1710.11089*, 2017.

Marlos C Machado, Marc G Bellemare, and Michael Bowling. Count-based exploration with the successor representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5125–5133, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Katharina Mülling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3): 263–279, 2013.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6292–6299. IEEE, 2018.

Kamal Ndousse, Douglas Eck, Sergey Levine, and Natasha Jaques. Multi-agent social reinforcement learning improves generalization. *arXiv preprint arXiv:2010.00581*, 2020.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 663–670, 2000.

Tom Le Paine, Sergio Gómez Colmenarejo, Ziyu Wang, Scott Reed, Yusuf Aytar, Tobias Pfaff, Matt W Hoffman, Gabriel Barth-Maron, Serkan Cabi, David Budden, et al. One-shot high-fidelity imitation: Training large-scale deep nets with rl. *arXiv preprint arXiv:1810.05017*, 2018.

Tom Le Paine, Caglar Gulcehre, Bobak Shahriari, Misha Denil, Matt Hoffman, Hubert Soyer, Richard Tanburn, Steven Kapturowski, Neil Rabinowitz, Duncan Williams, et al. Making efficient use of demonstrations to solve hard exploration problems. *arXiv preprint arXiv:1909.01387*, 2019.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Neural Information Processing Systems (NeurIPS)*, pp. 305–313, 1989.

Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Neil C Rabinowitz, Frank Perbet, H Francis Song, Chiyuan Zhang, SM Eslami, and Matthew Botvinick. Machine theory of mind. *arXiv preprint arXiv:1802.07740*, 2018.

Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3758–3765. IEEE, 2018.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 729–736, 2006.

Siddharth Reddy, Anca D Dragan, and Sergey Levine. Sqil: Imitation learning via reinforcement learning with sparse rewards. *arXiv preprint arXiv:1905.11108*, 2019.

Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. Deep imitative models for flexible inference, planning, and control. In *International Conference on Learning Representations (ICLR)*, April 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1134–1141. IEEE, 2018.

Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. *arXiv preprint arXiv:1810.07121*, 2018.

Kyriacos Shiarlis, Joao Messias, and SA Whiteson. Inverse reinforcement learning from failure. 2016.

Dale O Stahl. Evolution of smart-n players. *Games and Economic Behavior*, 5(4):604–617, 1993.

Freek Stulp, Gennaro Raiola, Antoine Hoarau, Serena Ivaldi, and Olivier Sigaud. Learning compact parameterized skills with a single regression. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 417–422. IEEE, 2013.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Matthew E Taylor, Halit Bener Suay, and Sonia Chernova. Integrating reinforcement learning with human demonstrations of varying ability. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 617–624, 2011.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Bernard Widrow and Fred W Smith. Pattern-recognizing control systems, 1964.

Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

Wako Yoshida, Ray J Dolan, and Karl J Friston. Game theory of mind. *PLoS Comput Biol*, 4(12): e1000254, 2008.

Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8. IEEE, 2018.

Jiangchuan Zheng, Siyuan Liu, and Lionel M Ni. Robust bayesian inverse reinforcement learning with sparse behavior noise. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A Related Work

Learning from demonstrations. Learning from demonstrations, also referred to as imitation learning (IL, Widrow & Smith, 1964; Pomerleau, 1989; Atkeson & Schaal, 1997), is an attractive framework for sequential decision making when reliable, expert demonstrations are available. Early work on IL assumed access to high-quality demonstrations and aimed to match the expert policy (Pomerleau, 1991; Heskes, 1998; Ng et al., 2000; Abbeel & Ng, 2004; Billard et al., 2008; Argall et al., 2009; Ziebart et al., 2008). Building on this assumption, many recent works have studied various aspects of both *single-task* (Ratliff et al., 2006; Wulfmeier et al., 2015; Choi & Kim, 2011; Finn et al., 2016b; Ho & Ermon, 2016; Finn et al., 2016a; Fu et al., 2017; Zhang et al., 2018; Rahmatizadeh et al., 2018) and *multi-task* (Dimitrakakis & Rothkopf, 2011; Mülling et al., 2013; Stulp et al., 2013; Deisenroth et al., 2014; Sharma et al., 2018; Codevilla et al., 2018; Fu et al., 2019; Rhinehart et al., 2020; Filos et al., 2020) IL. However, these methods are all limited by the performance of the demonstrator that they try to imitate. Learning from suboptimal demonstrations has been studied by Coates et al. (2008); Grollman & Billard (2011); Zheng et al. (2014); Choi et al. (2019); Shiarlis et al. (2016); Brown et al. (2019), enabling, under certain assumptions, imitation learners to surpass their demonstrators' performance. In contrast, our method integrates demonstrations into an online reinforcement learning pipeline and can use the demonstrations to improve learning on a new task.

Reinforcement learning with demonstrations. Demonstration trajectories have been used to accelerate the learning of RL agents (Taylor et al., 2011; Vecerik et al., 2017; Rajeswaran et al., 2017; Hester et al., 2018; Gao et al., 2018; Nair et al., 2018; Paine et al., 2018; 2019), as well as demonstrations where actions and/or rewards are unknown (Borsa et al., 2017; Torabi et al., 2018; Sermanet et al., 2018; Liu et al., 2018; Aytar et al., 2018; Brown et al., 2019). In contrast to the standard imitation learning setup, these methods allow improving over the expert performance as the policy can be further fine-tuned via reinforcement learning. Offline reinforcement learning with online fine-tuning (Kalashnikov et al., 2018; Levine et al., 2020) can be framed under this settings too. Our method builds on the same principles, however, unlike these works, we do not assume that the demonstration data either come with reward annotations, or that they relate to the same task the RL agent is learning (i.e., we learn from multi-task demonstrations which may include irrelevant tasks).

Successor features. Successor features (SFs) are a generalisation of the successor representation (Dayan, 1993) for continuous state and action spaces (Barreto et al., 2017). Prior work has used SFs for (i) zero-shot transfer (Barreto et al., 2017; Borsa et al., 2018; Barreto et al., 2020); (ii) exploration (Janz et al., 2019; Machado et al., 2020); (iii) skills discovery (Machado et al., 2017; Hansen et al., 2019); (iv) hierarchical RL (Barreto et al., 2019) and theory of mind (Rabinowitz et al., 2018). Nonetheless, in all the aforementioned settings, direct access to the rewards or cumulants was provided. Our method, instead, uses demonstrations without reward labels for inferring the cumulants and learning the corresponding SFs. More closely to this work, Lee et al. (2019) propose learning cumulants and successor features for a *single-task* IRL setting. Their approach differs from ours in two key respects: first, they use a learned dynamics model to learn the cumulants. Second, the learned SFs are used for representing the action-value function, not to inform the behaviour policy with GPI.

Model of others in multi-agent learning. Our method draws inspiration and builds on the multi-agent learning setting, where multiple agents participate in the same environment and the states, actions of others are observed (Davidson, 1999; Lockett et al., 2007; He et al., 2016; Jaques et al., 2019). However, we do not explore *strategic* settings, where recursive reasoning (Stahl, 1993; Yoshida et al., 2008) is necessary for optimal behaviour.

B EXPERIMENTAL DETAILS

In this section we describe the environments used in our experiments (see Section 4) and the experiment design.

B.1 HIGHWAY

We build on the `highway-v0` task from the `highway-env` traffic simulator (Leurent, 2018). The task is specified by:

1. **State space, $\mathcal{S}$:** The kinematic information of the ego vehicle and the five closest vehicles (ordered from closest to the furthest) is used as the Markov state, i.e., $\mathbf{s}_t = \{[x_t, y_t, \dot{x}_t, \dot{y}_t]\}_{\text{ego, other}_1, \ldots, \text{other}_5} \in \mathbb{R}^{6\times 4}$. The ego-car is illustrated in green and the other cars in blue.
2. **Action space, $\mathcal{A}$:** We use a discrete action space, constructed by K-means clustering of the continuous actions of the intelligent driving model (Kesting et al., 2010). We found out that keeping 9 actions was sufficient, i.e., $\mathbf{a}_t \in \{0, \ldots, 8\}$.
3. **Demonstrations, $\mathcal{D}$:** At each time-step, the ego-car observes online the state-action pairs for the 5 closest cars.

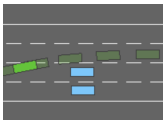


Figure 2: Highway

B.2 ROUNDABOUT

We build on the `roundabot-v0` task from the `highway-env` traffic simulator (Leurent, 2018). The task is specified by:

1. **State space, $\mathcal{S}$:** The kinematic information of the ego vehicle and the five closest vehicles (ordered from closest to the furthest) is used as the Markov state, i.e., $\mathbf{s}_t = \{[x_t, y_t, \dot{x}_t, \dot{y}_t]\}_{\text{ego, other}_1, \ldots, \text{other}_3} \in \mathbb{R}^{4\times 4}$. The ego-car is illustrated in green and the other cars in blue.
2. **Action space, $\mathcal{A}$:** We use a discrete action space, constructed by K-means clustering of the continuous actions of the intelligent driving model (Kesting et al., 2010). We found out that keeping 6 actions was sufficient, i.e., $\mathbf{a}_t \in \{0, \ldots, 5\}$.
3. **Demonstrations, $\mathcal{D}$:** At each time-step, the ego-car observes online the state-action pairs for the 3 closest cars.

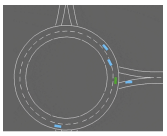


Figure 3: Roundabout

B.3 COINGRID

We build a simple multi-task grid-world. The task is specified by:

1. **State space, $\mathcal{S}$:** We use a symbolic, multi-channel representation of the 7×7 gridworld (Chevalier-Boisvert et al., 2019): the first three channels specify the presence or absence of the three different coloured boxes, the forth channel was the walls mask and the fifth and last channel was the position and orientation of the agent. We represent the orientation of the agent by 'painting' the cell in front of the agent. Therefore $\mathbf{s}_t \in \{0, 1\}^{7\times 7\times 5}$.
2. **Action space, $\mathcal{A}$:** We use the {`LEFT`, `RIGHT`, `FORWARD`} actions from Minigrid (Chevalier-Boisvert et al., 2018) to navigate the maze, i.e., $\mathbf{a}_t \in \{0, 1, 2\}$.
3. **Demonstrations, $\mathcal{D}$:** At the beginning of training, the agent is given state-action pairs of other agents collecting either red or green coins.

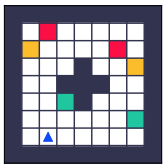


Figure 4: CoinGrid

B.4 FRUITBOT

We build on the `Fruitbot` environment from OpenAI's ProcGen benchmark (Cobbe et al., 2020). The task is specified by:

1. **State space, $\mathcal{S}$:** We use the original high-dimensional 64×64 RGB observations, i.e., $\mathbf{s}_t \in [0, 1]^{64 \times 64 \times 3}$.
2. **Action space, $\mathcal{A}$:** We use the original 15 discrete actions, i.e., $\mathbf{a}_t \in \{0, \dots, 14\}$.
3. **Demonstrations, $\mathcal{D}$:** At each time-step, the agent observes online the states and actions of 3 trained agents playing the game in parallel: One agent collects both fruits and other objects, one collects other objects and avoids fruits and the last one randomly selects actions.

Figure 5: Fruitbot

C Implementation Details

For our experiments we used Python (Van Rossum & Drake Jr, 1995). We used JAX (Bradbury et al., 2018; Babuschkin et al., 2020) as the core computational library, Haiku (Hennigan et al., 2020) and Acme (Hoffman et al., 2020) for implementing $\Psi\Phi$-learning and the baselines, see Section 4. We also used Matplotlib (Hunter, 2007) for the visualisations and Weightd & Biases (Biewald, 2020) for managing the experiments.

C.1 Computation Graph

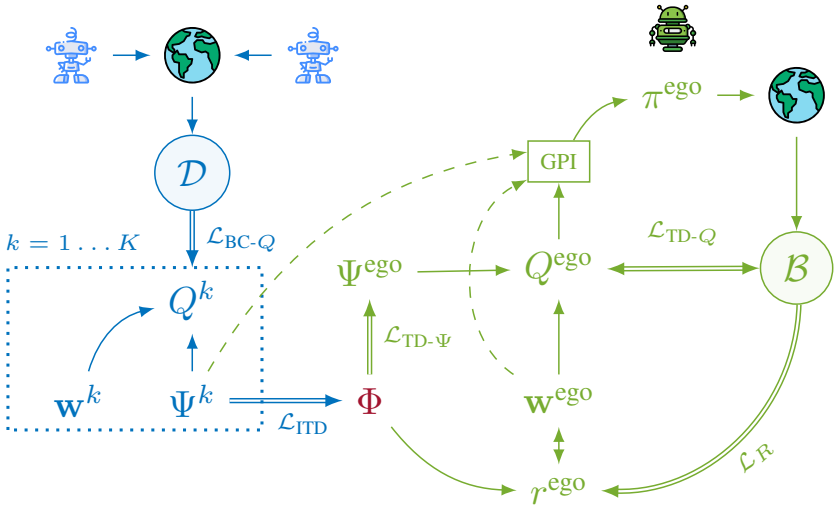


Figure 6: **Computational graph of the $\Psi\Phi$-learning algorithm.** Demonstrations $\mathcal{D}$ contain data from other agents for *unknown* tasks. We employ *inverse temporal difference learning* (ITD, see Section 3.1) to recover other agents' successor features (SFs) and preferences. The ego-agent combines the estimated SFs of others along with its own preferences and successor features with generalised policy improvement (GPI, see Section 2.2), generating experience. Both the demonstrations and the ego-experience are used to learn the shared cumulants. Losses $\mathcal{L}_*$ are represented with double arrows and gradients flow according to the pointed direction(s).

C.2 Neural Network Architecture

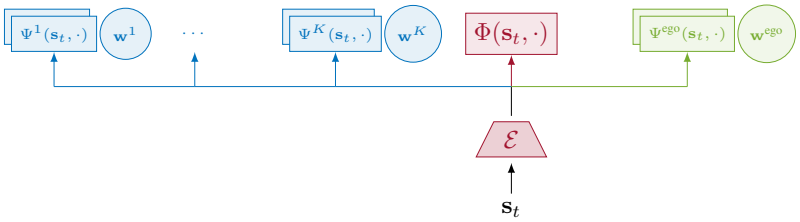


Figure 7: **Neural network architecture of the $\Psi\Phi$-learner.** The rectangular nodes are tensors parametrised by MLPs and the circles are learnable vectors. We share an observation network/torso, $\mathcal{E}$, across all the network heads. The network heads that related to the other agents are in blue and trained from demonstrations $\mathcal{D}$. The ego-agent's experience $\mathcal{B}$ is used for training the green heads. The shared cumulants and torso are trained with both $\mathcal{D}$ and $\mathcal{B}$. An ensemble of two successor features approximators is used for the ego- and other- agents for combatting model overestimation, see Section 3.

Table 2: **$\Psi\Phi$-learner's hyperparameters per environment**. The tuning was performed on a DQN (Mnih et al., 2013) baseline with population based training (Jaderberg et al., 2017) using Weights & Biases (Biewald, 2020) integration with Ray Tune (Liaw et al., 2018). We selected the best hyperparameters configuration out of 32 trials per environment and used this for our $\Psi\Phi$-learner.

	`Highway`	CoinGrid	`FruitBot`
Torso network, $\mathcal{E}$	MLP([512, 256])	IMPALA (Espeholt et al., 2018), shallow (no LSTM)	IMPALA (Espeholt et al., 2018), deep (no LSTM)
Cumulants approximator, Φ	MLP([128, 128])	MLP([256, 128])	MLP([256, 128])
Successor features approximator, Ψ	MLP([256, 128])	MLP([512, 256])	MLP([512, 256])
Ensemble size, Ψ	2	2	2
$\mathcal{L}_1$ coefficient	0.05	0.05	0.05
Number of dimensions in Φ	8	4	64
Minibatch size	512	64	32
n-step	4	8	128
Discount factor, γ	1.0	0.9	0.999
Target network update period	100	1000	2500
Optimiser	ADAM (Kingma & Ba, 2014), `lr=1e-3`	ADAM (Kingma & Ba, 2014), `lr=1e-4`	ADAM (Kingma & Ba, 2014), `lr=5e-5`

C.3 Hyperparameters

C.4 Compute Resources

All the experiments were run on Microsoft Azure `Standard_NC6s_v3` machines, i.e., with a 6-core vCPU, 112GB RAM and a single NVIDIA Tesla V100 GPU. The iteration cycle for (i) **Highway** experiments was 3 hours; (ii) **CoinGrid** experiments was 5.5 hours and (iii) **Fruitbot** experiments was 19 hours.

D PROOFS

We begin by formalizing the statement of Theorem 1. When not specified the norm $\|\cdot\|$ refers to the 2-norm. Given a function $F : \mathcal{X} \to \mathbb{R}^d$ for some finite set $\mathcal{X}$, we will write $F(x)$ to denote the value of the function on input x and F to denote the matrix representation of this function in $\mathbb{R}^{|\mathcal{X}| \times d}$.

Theorem 1 (Formal statement)**.** *Let $\mathcal{C} = (\mathcal{S}, \mathcal{A}, P, \gamma)$ be a CMP with a finite state space. Let $\phi : \mathcal{S} \to \mathbb{R}^d$, and let $\Phi = \phi(\mathcal{S}) \in \mathbb{R}^{|\mathcal{S}| \times d}$. Let $(r_i)_{i=1}^k$ denote a set of reward functions on $\mathcal{C}$, $\tilde{\Psi}^i$ be a collection of successor features approximations for policies $(\pi^i)_{i=1}^k$ (π_i optimal for r_i) with true successor feature values Ψ^i, and w_i the best least-squares linear approximator of r_i given Φ, with errors*

$$\|\Phi w_i - r_i\|_\infty < \delta_r \text{ and } \|\tilde{\Psi}^i - \Psi^i\| < \delta_\Psi \quad \forall i.$$

Let w' be a new preference vector for a reward function r', with maximal error δ_r as well. Let $\tilde{Q}^i = \tilde{\Psi}^i w'$. Let π^* be the optimal policy for the ego task w' and let π be the GPI policy obtained from $\{\tilde{Q}^{\pi_i}\}$, with δ_r, δ_Ψ the reward and successor feature approximation errors. Then for all s, a

$$Q^*(s,a) - Q^\pi(s,a) \leq \frac{2}{1-\gamma}\left[(\phi_{\max}\|w_j - w'\| + 2\delta_r) + \|w'\|\delta_\Psi + \frac{1}{(1-\gamma)}\delta_r\right] \tag{15}$$

Barreto et al. (2017) construct their bound on the sub-optimality of the GPI policy as a function of the error of the value approximations $\widetilde{Q}^i$. Because we bound the reward approximation error, rather than the value approximation error, we require an additional step to obtain a bound on the errors of the value funciton approximations. To prove Theorem 1, we must therefore first use the following lemma to bound the effect of the *reward approximation error* on the value approximation error. While this result is straightforward, we include a short proof for completeness.

Lemma 1. *Fix some policy π. Let r be reward vector and let w be the least-squares solution to $\min \|\Phi w - r\|$. Let Ψ^π be the true successor features for Φ under policy π, and let Q^π be the value. Let $\delta_r = R(\mathcal{S}) - \Phi w$, $\delta_{max} = \|\delta_r\|_\infty$. Then letting $\tilde{Q} = \Psi w$, we have*

$$\|Q^\pi - \tilde{Q}\|_\infty \leq \frac{1}{1-\gamma}\delta_r \tag{16}$$

Proof.

$$\|Q^\pi - \tilde{Q}\|_\infty \leq \sum \gamma^t \|P^{\pi t}(\Phi w - r)\|_\infty \tag{17}$$

$$\leq \sum \gamma^t \|P^{\pi t}\delta_r\|_\infty = \sum_t \gamma^t \max_{s'} |\sum_{s \in \mathcal{S}} (P^\pi)^t(s', s)\delta_r(s)| \tag{18}$$

Since P^π is a stochastic matrix, so are all of its powers, and so the rows of $(P^\pi)^t$ sum to 1.

$$\leq \sum_t \gamma^t \max_{s'} |\sum P^{\pi\, t}(s, s')\delta_{\max}| = \sum \gamma^t \delta_{\max} \tag{19}$$

$$= \frac{1}{1-\gamma}\delta_{\max} \tag{20}$$

□

We now prove the main result.

Proof. We follow the proof of Barreto et al. (2017, Theorem 2), with additional error terms to account for the reward and successor feature approximation errors.

$$
\begin{aligned}
Q^*(s,a) - Q^{\pi}(s,a) &\leq Q^*(s,a) - Q^{\pi_j}(s,a) + \frac{2}{1-\gamma}\epsilon && \text{(Barreto et al., 2017, Theorem 1)} \\
&\leq \frac{2}{1-\gamma}\|r_j - r'\|_\infty + \frac{2}{1-\gamma}\epsilon && \text{(Barreto et al., 2017, Lemma 1)} \\
&\leq \frac{2}{1-\gamma}\|\phi w_j + \delta_j - \phi w' - \delta'\|_\infty + \frac{2}{1-\gamma}\epsilon \\
&\leq \frac{2}{1-\gamma}(\phi_{\max}\|w_j - w'\| + \delta_r + \delta_r) + \frac{2}{1-\gamma}\epsilon \\
&\leq \frac{2}{1-\gamma}(\phi_{\max}\|w_j - w'\| + 2\delta_r) + \frac{2}{1-\gamma}\|\tilde{\Psi}^j w' - \Psi_j w' + \Psi_j w' - Q_j\| \\
&\leq \frac{2}{1-\gamma}(\phi_{\max}\|w_j - w'\| + 2\delta_r) + \frac{2}{1-\gamma}\|\tilde{\Psi}^j w' - \Psi_j w'\| + \|\Psi_j w' - Q_j\| \\
&\leq \frac{2}{1-\gamma}(\phi_{\max}\|w_j - w'\| + 2\delta_r) + \frac{2}{1-\gamma}\|w'\|\delta_\Psi + \frac{2}{1-\gamma}\|\Psi_j w' - Q_j\| \\
&\leq \frac{2}{1-\gamma}(\phi_{\max}\|w_j - w'\| + 2\delta_r) + \frac{2}{1-\gamma}\|w'\|\delta_\Psi + \frac{2}{1-\gamma}(\frac{1}{1-\gamma}\delta_r) && \text{(Lemma 1)} \\
&= \frac{2}{1-\gamma}\left[(\phi_{\max}\|w_j - w'\| + 2\delta_r) + \|w'\|\delta_\Psi + \frac{1}{(1-\gamma)}\delta_r\right]
\end{aligned}
$$

□

E ALGORITHMS

Algorithm 1: Inverse Temporal Difference Learning

Input :

$\mathcal{D} = \{(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{a}_T; k)_{k=1}^K\}$	No-reward demonstrations
$\lambda_{\mathbf{w}}$	$\mathcal{L}_1$ loss coefficient

Output :

$\boldsymbol{\theta}_\Phi$	Parameters of cumulants network
$\{\boldsymbol{\theta}_{\Psi^k}\}_{k=1}^K$	Parameters of successor features approximators
$\{\mathbf{w}^k\}_{k=1}^K$	Preferences vectors for the K agents

// initialisations

Initialise parameters $\boldsymbol{\theta}_\Phi, \{\boldsymbol{\theta}_{\Psi^k}, \mathbf{w}^k\}_{k=1}^K$

while *budget* **do**

- Sample trajectories $\{\tau_i = (s_1^{(i)}, a_1^{(i)}, \ldots, s_T^{(i)}, a_T^{(i)}; k^{(i)})\}_{i=1}^N \sim \mathcal{D}$
- Calculate behavioural cloning loss $\mathcal{L}_{\text{BC-}Q}(\boldsymbol{\theta}_{\Psi^k}, \mathbf{w}^k)$ on samples $\{\tau_i\}_{i=1}^N$ ▷ see Eqn. (8)
- $\boldsymbol{\theta}_{\Psi^k} \overset{\alpha}{\leftarrow} \nabla_{\boldsymbol{\theta}_{\Psi^k}} \mathcal{L}_{\text{BC-}Q}(\boldsymbol{\theta}_{\Psi^k}, \mathbf{w}^k)$ ▷ update Ψs
- $\mathbf{w}^k \overset{\alpha}{\leftarrow} \nabla_{\mathbf{w}^k} \left(\mathcal{L}_{\text{BC-}Q}(\boldsymbol{\theta}_{\Psi^k}, \mathbf{w}^k) + \lambda_{\mathbf{w}} \|\mathbf{w}^k\|_1\right)$ ▷ update $\mathbf{w}$s
- Calculate inverse temporal difference loss $\mathcal{L}_{\text{ITD}}(\boldsymbol{\theta}_\Phi)$ on samples $\{\tau_i\}_{i=1}^N$ ▷ see Eqn. (9)
- $\boldsymbol{\theta}_\Phi \overset{\alpha}{\leftarrow} \nabla_{\boldsymbol{\theta}_\Phi} \mathcal{L}_{\text{TD-}\Psi}(\boldsymbol{\theta}_{\Psi_k}, \boldsymbol{\theta}_\Phi)$ ▷ update Φ

Algorithm 2: $\Psi\Phi$-Learning

Input :

$\mathcal{D} = \{(\mathbf{s}_1, \mathbf{a}_1, \dots, \mathbf{a}_T; k)_{k=1}^K\}$ No-reward demonstrations

$\lambda_{\mathbf{w}}$ $\mathcal{L}_1$ loss coefficient

Output :

$\boldsymbol{\theta}_{\Psi^{\text{ego}}}$ Ego successor features approximator

$\boldsymbol{\theta}_{\Phi}$ Parameters of cumulants network

$\{\boldsymbol{\theta}_{\Psi^k}\}_{k=1}^K$ Parameters of successor features approximators

$\{\mathbf{w}^k\}_{k=1}^K$ Preferences vectors for the K agents

`// initialisations`

Empty replay buffer for ego-experience $\mathcal{B} = \{\}$

Initialise parameters $\boldsymbol{\theta}_{\Psi^{\text{ego}}}, \mathbf{w}^{\text{ego}}, \boldsymbol{\theta}_{\Phi}, \{\boldsymbol{\theta}_{\Psi^k}, \mathbf{w}^k\}_{k=1}^K$

while *budget* **do**

`// agent-environment interaction`

Reset episode, $\mathbf{s} \leftarrow$ `env.reset()`, $t \leftarrow 0$

while *not done* **do**

$\mathbf{w}^{\text{ego}} \leftarrow \arg\min_w \mathcal{L}_{\text{R}}(\boldsymbol{\theta}_{\Phi}, w; \mathcal{B})$ ▷ `ego-task inference, see Eqn. (10)`

$\mathbf{a} \leftarrow \pi^{\text{ego}}_{\text{GPI}}\left(\mathbf{s}; \Psi^{\text{ego}}, \mathbf{w}^{\text{ego}}, \{\boldsymbol{\theta}_{\Psi^k}\}_{k=1}^K\right)$ ▷ `GPI, see Eqn. (13)`

Step in the environment, $\mathbf{s}', r^{\text{ego}},$ `done` $\leftarrow$ `env.step(`$\mathbf{a}$`)`

Append transition in the replay buffer, $\mathcal{B} \leftarrow \mathcal{B} \cup (\mathbf{s}, \mathbf{a}, r^{\text{ego}}, \mathbf{s}')$

$\mathbf{s}' \leftarrow \mathbf{s}, t \leftarrow t + 1$

(online demonstrations) Append demonstrations in $\mathcal{D}$ ▷ `optional`

`// parameter updates/learning`

$\boldsymbol{\theta}_{\Phi}, \{\boldsymbol{\theta}_{\Psi^k}, \mathbf{w}^k\}_{k=1}^K \leftarrow$ `ITD` $\left(\mathcal{D}, \lambda_{\mathbf{w}}, \boldsymbol{\theta}_{\Phi}, \{\boldsymbol{\theta}_{\Psi^k}, \mathbf{w}^k\}_{k=1}^K\right)$ ▷ `see Algorithm. (1)`

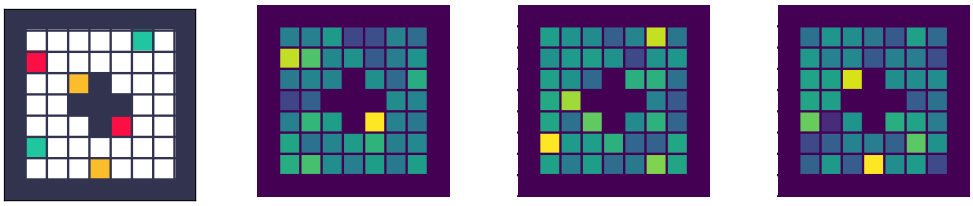
Sample transitions $\{(\mathbf{s}^{(i)}, \mathbf{a}^{(i)}, r^{\text{ego},(i)}, \mathbf{s}'^{(i)})\} \sim \mathcal{B}$

Calculate the reward loss $\mathcal{L}_{\text{R}}(\boldsymbol{\theta}_{\Phi}, \mathbf{w}^{\text{ego}})$ ▷ `see Eqn. (10)`

$\boldsymbol{\theta}_{\Phi} \overset{\alpha}{\leftarrow} \nabla_{\boldsymbol{\theta}_{\Phi}} \mathcal{L}_{\text{R}}(\boldsymbol{\theta}_{\Phi}, \mathbf{w}^{\text{ego}})$ ▷ `update` Φ

Calculate TD losses $\mathcal{L}_Q(\boldsymbol{\theta}_{\Psi^{\text{ego}}})$ and $\mathcal{L}_{\text{TD-}\Psi}(\boldsymbol{\theta}_{\Psi^{\text{ego}}})$ ▷ `see Eqn. (11,12)`

$\boldsymbol{\theta}_{\Psi^{\text{ego}}} \overset{\alpha}{\leftarrow} \nabla_{\boldsymbol{\theta}_{\Psi^{\text{ego}}}} \left(\mathcal{L}_Q(\boldsymbol{\theta}_{\Psi^{\text{ego}}}) + \frac{1}{|\Psi|}\mathcal{L}_{\text{TD-}\Psi}(\boldsymbol{\theta}_{\Psi^{\text{ego}}})\right)$ ▷ `update` Ψ^{ego}



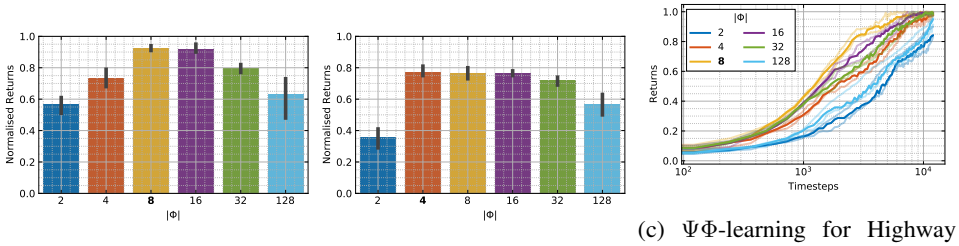
(a) CoinGrid (b) $\hat{\phi}_1$ (c) $\hat{\phi}_2$ (d) $\hat{\phi}_3$

Figure 8: Qualitative evaluation of the learned cumulants in the CoinGrid task. Cumulants $\hat{\phi}_1$, $\hat{\phi}_2$, and $\hat{\phi}_3$ seem to capture the red, green, and yellow blocks, respectively. Therefore, linear combinations of the learned cumulants can represent arbitrary rewards in the environment, which involve stepping on the coloured blocks.

F VISUALISATIONS

(a) CoinGrid (b) ϕ_1 (c) ϕ_2 (d) ϕ_3 (e) ϕ_4

Figure 9: Qualitative evaluation of the learned cumulants in the CoinGrid task. Cumulants ϕ_1, ϕ_2, and ϕ_3 seem to capture the red, green, and yellow blocks, respectively. The yellow blocks are captured by both and ϕ_4. Therefore, linear combinations of the learned cumulants can represent arbitrary rewards in the environment, which involve stepping on the coloured blocks.



(a) ITD for Roundabout (b) ITD for CoinGrid (c) $\Psi\Phi$-learning for Highway Multi-Task

Figure 10: Sensitivity of our ITD (see Section 3.1) and $\Psi\Phi$-learning (see Section 3.2) algorithms to the dimensionality of the learned cumulants. We consistently observe across all three experiments (a)-(c) that for a small number of Φ dimensions the cumulants are not expressive enough to capture the axis of variation of the different agents' reward functions (including the ego-agent in (c)). We also note that the performance of both ITD and $\Psi\Phi$-learning is relative robust for a medium and large number of Φ dimensions. We attribute this to the used sparsity prior, i.e., $\mathcal{L}_1$ loss, to the preferences $\mathbf{w}$. In our experiments we selected the smallest number of Φ dimensions that demonstrated good performance to keep the number of model parameters as small as possible (in bold in the figures and reported in Table 2).

G INVERSE TEMPORAL DIFFERENCE LEARNING: EMPIRICAL RESULTS

Table 3: We evaluate how well ΨΦ-learning is able to infer the correct reward function by training an RL agent on the inferred rewards, and comparing this to alternative imitation learning methods in three environments. All methods are trained on expert demonstrations. A "♢" indicates methods that infer an *explicit* reward function and then use one of DQNor PPOto train an RL agent, depending on the environment. A "♣" indicates methods that directly learn a policy from demonstrations. A "†" indicates methods that use privileged task id information for handling multi-task demonstrations. We report mean and standard error of *normalised returns* over 3 runs, where higher-is-better and the performance is upper bounded by 1.0, reached by the same RL agent, trained with the ground truth reward function.

Methods	`Roundabout`$^{\text{DQN}}$	CoinGrid $^{\text{DQN}}$	`FruitBot`$^{\text{PPO}}$
BC$^{\dagger\clubsuit}$ (Pomerleau, 1989)	0.81±0.02	0.69±0.06	**0.37**±0.02
SQIL$^{\dagger\clubsuit}$ (Reddy et al., 2019)	0.85±0.02	0.64±0.05	**0.35**±0.03
GAIL$^{\dagger\diamondsuit}$ (Ho & Ermon, 2016)	0.77±0.07	0.73±0.02	0.31±0.02
ITD$^{\diamondsuit}$ (ours, cf. Section 3.1)	**0.92**±0.01	**0.77**±0.03	**0.35**±0.04

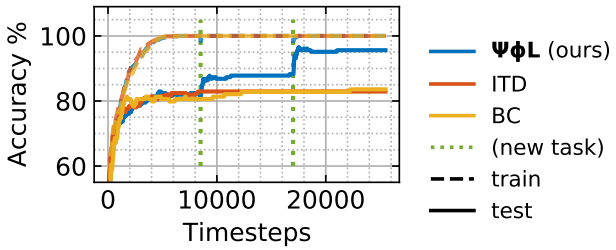


Figure 11: Test accuracy in predicting other agents' actions.The shared cumulants Φ for modelling others- and ego- reward functions allow our ΨΦ-learner to improve its ability to predict others' actions by experiencing new ego-tasks. Pure imitation learning and our ITD inverse RL methods achieve high train accuracy but they do not have a mechanism for utilising RL experience to improve their generalisation to the test set as new tasks are provided.