PsiPhi-Learning: Reinforcement Learning with Demonstrations using Successor Features and Inverse Temporal Difference Learning

Angelos Filos
Clare Lyle
Yarin Gal
Sergey Levine
Natasha Jaques
Gregory Farquhar

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 ΨΦ-learning (pronounced ‘Sci-Fi’). We provide empirical evidence for the effectiveness of ΨΦ-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 naive 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: \( C \triangleq (S, A, P, \gamma) \). \( S \) and \( A \) represent the continuous and discrete action spaces, respectively, \( s' \sim P(\cdot|s, a) \) describes the transition dynamics and \( \gamma \) is the discount factor.

A task is formulated as a Markov decision process (MDP, Puterman, 2014), characterised by a reward function, \( R: S \times A \rightarrow \mathbb{R} \), i.e., \( M \triangleq (C, R) \).

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

### 2.1. RL with No-Reward Demonstrations

We are interested in settings in which, in addition to an environment \( C \), 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 \( D = \{ \tau_1, \tau_2, \ldots, \tau_N \} \), where the trajectory \( \tau \triangleq (s_0, a_0, \ldots, s_T, a_T; k) \) 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 \( B = \{(s, a, s', r^{\text{epo})})\} \). Due to the lack of reward annotations in \( D \), and the fact that the data may be irrelevant to the
ego-agent’s task, it is not trivial to combine demonstrations from \( D \) with the ego-agent’s experience \( B \).

### 2.2. Successor Features and Cumulants

To make use of the demonstrations \( D \), 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 \( \Phi(s, a) \in \mathbb{R}^d \), and task-specific preferences \( w \in \mathbb{R}^d \):

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

The preferences \( w \) are a representation of a possible goal in the world \( C \), in the sense that each \( w \) gives rise to a task \( M^w = (C, R^w) \). We use ‘task’, ‘goal’, and ‘preferences’ interchangeably when context makes it clear whether we are referring to \( w \) itself, or the corresponding \( M^w \) or \( R^w \).

The action-value function for a policy \( \pi \) in \( M^w \) is then a function of the preferences \( w \) and the \( \pi \)'s successor features.

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

\[
\Psi^\pi(s, a) \triangleq \mathbb{E}^{C, \pi}[\sum_{t=0}^{\infty} \gamma^t \Phi(s_t, a_t) \mid s_0 = s, a_0 = a]. \tag{2}
\]

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

An action-value function is then given by the dot product of the preferences \( w \) and \( \pi \)'s SFs:

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

**Proof.** See (Barreto et al., 2017). \( \square \)

Note that if we have \( \Psi^\pi \), the value of \( \pi \) for a new preference \( w' \) 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 \( \pi' \) s.t.

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

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

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

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, shown in Figure 1. 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, \( D \), we model the agents that generated the data (i.e., blue nodes in Figure 1) as soft-optimal for an unknown task. In particular, the \( k \)-th agent’s policy is soft-optimal under task \( w^k \) and is given by

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

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 \( \Phi \). The \( k \)-th agent’s successor features are temporally consistent with these cumulants \( \Phi \):

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

To learn these quantities from \( K \) demonstrators, we parameterise the SFs with \( \theta^\Psi \), preferences with \( w^k \), and shared cumulants with \( \theta_\Phi \). A schematic of the model architecture and further details are provided in Appendix B. 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., \( D^k \subseteq D \), we train its successor features \( \theta^\Psi \) and the preferences \( w^k \) by minimising
the negative log-likelihood of the demonstrations
\[
\mathcal{L}_{BC-Q}(\theta_{\Psi}, w^k) \triangleq \mathbb{E}_{\tau \sim D} \log \frac{\exp(\Psi(s_t, a_t; \theta_{\Psi}^k) \cdot w^k)}{\sum_{a_t'} \exp(\Psi(s_t, a_t'; \theta_{\Psi}^k) \cdot w^k)}. \tag{8}
\]
Importantly, Eqn. (8) reflects the fact that along a trajectory \(\tau\), the successor features are a function of the state and action at each time-step, \(s_t\) and \(a_t\), while the preferences \(w^k\) are learnable but consistent across time and trajectories. The direction of the preference \(w^k\) indicates the goal of the agent \(k\) by showing how relatively rewarding it finds the different dimensions of the cumulant features \(\Phi\). The learned magnitude of \(w^k\) can further capture how greedily the agent \(k\) pursues this goal.

A sparsity prior, i.e., \(\mathcal{L}_1\) loss, on preferences \(w^k\) is also used to promote disentangled cumulant dimensions (see Figure 4). We found the \(\mathcal{L}_1\) loss made the algorithm more robust to the choice of dimension of \(\Phi\) (see Figure 13 in the Appendix), but did not substantially affect overall performance.

**Inverse temporal difference loss.** The cumulant parameters \(\theta_{\Phi}\) 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 \(\Phi\). 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 \(\theta_{\Phi}\), \(\theta_{\Psi}^k\) to be (self-)consistent with these SFs by minimising

\[
\mathcal{L}_{ITD}(\theta_{\Phi}, \theta_{\Psi}^k) \triangleq \mathbb{E}_{(s_t, a_t, s_{t+1}, a_{t+1}, k) \sim D} \left\| \Psi(s_t, a_t; \theta_{\Psi}^k) \leftrightarrow -\Phi(s_t, a_t; \theta_{\Phi}) - \gamma \Psi(s_{t+1}, a_{t+1}; \theta_{\Psi}^k) \right\|_2. \tag{9}
\]
In practice, our ITD-learning algorithm alternates between minimising \(\mathcal{L}_{BC-Q}\) for training only the successor features and preferences, and \(\mathcal{L}_{ITD}\) for training both the shared cumulants and successor features, provided only with no-reward demonstrations, as illustrated in Figure 1.

From the definition of cumulants and preferences, we can recover the \(k\)-th demonstrator’s reward, by applying Eqn. (1), i.e., \(R^k(s, a) \approx \Psi(s, a; \theta_{\Psi}^k) \cdot w^k\). 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.

**Theorem 1** (Validity of the ITD Minimiser). The minimisers of \(\mathcal{L}_{BC-Q}\) and \(\mathcal{L}_{ITD}\) are potentially-shaped cumulants that explain the observed reward-free demonstrations.

**Proof.** See Appendix C.

\(^1\)Hence the name inverse TD learning.

**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 of preferences, so we can use \(w = 1\). In this case \(\Psi\) is the action-value function \(Q\) and \(\Phi\) 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 \(\Phi\) 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 depicted in Figure 1 and 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, \(\Psi_ego\), and preferences, \(w_ego\), as in Eqn. (3). The ego-agent interacts with the environment, storing its experience in a replay buffer, \(B \leftarrow B \cup \{(s, a, s', r_ego)\}\). The \(\Psi\Phi\)-learner also has estimates for the cumulants, per-agent SFs and preferences obtained with ITD from the demonstrations \(D\).

**Reward loss.** The ego-rewards, \(r_ego\), are used to ground the cumulants \(\Phi\) and the preferences \(w_ego\), via the loss

\[
\mathcal{L}_R(\theta_{\Phi}, w_ego) \triangleq \mathbb{E}_{(a, a', r_ego) \sim B} \left\| \Psi(s, a; \theta_{\Phi})^\top w_ego - r_ego \right\|. \tag{10}
\]

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 \(\Phi\), 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

\[
\mathcal{L}_{Q}(\theta_{\Psi_ego}) \triangleq \mathbb{E}_{(s, a, a', r_ego) \sim B} \left\| \Psi(s, a; \theta_{\Psi_ego})^\top w_ego \leftrightarrow -r_ego - \gamma \max_{a'} \Psi(s', a'; \theta_{\Psi_ego})^\top w_ego \right\|. \tag{11}
\]
We additionally train the successor features to be self-consistent (i.e. to satisfy Equation 2) using a TD loss $L_{TD-\Phi}$.

$$L_{TD-\Phi}(\theta_{\Psi^{\Phi}}) \triangleq \mathbb{E}_{(s,a,a',r) \sim B}(\|\hat{\Psi}(s,a;\theta_{\Psi^{\Phi}}) - \hat{\Phi}(s,a;\theta_{\Psi^{\Phi}}) - \gamma \Psi(s',a';\theta_{\Psi^{\Phi}})\|).$$

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

$$\pi^{ego}(s) = \arg \max_a \max_{\theta_{\Psi}} \Psi(s,a;\theta_{\Psi})^\top w^{ego}.$$  

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 $\{\Psi^*\}_{k=1}^K$, 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 $\hat{Q}^* = \Psi^* w'$. We extend this result to explicitly account for the reward approximation error $\delta_r$ obtained by the the learned cumulants and the SF approximation error $\delta_{\Psi}$.

**Theorem 2** (Generalisation Bound of $\Psi-\Phi$-Learning). Let $\pi^*$ be the optimal policy for the ego task $w'$ and let $\pi$ be the GPI policy obtained from $\{\hat{Q}^\pi\}$, with $\delta_r, \delta_{\Psi}$ the reward and successor feature approximation errors. Then $\forall s, a$

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

**Proof.** See Appendix C 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 2 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 $\rho(a|s)$ 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 2. 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 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. red. 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
Figure 2. Environments studied in this paper. Environments (a-b) are multi-agent environments in which the ego-agent must learn online from other agents, and learn to navigate around other agents in the environment (see Section 4.2). Environments (c-d) are single-agent. In environment (c) we test whether the ego-agent can learn offline from a set of demonstrations previously collected by other agents (see Section 4.5). Environment (d) is used to test whether our method can scale to more complex, high-dimensional tasks (see Section 4.2).

4.2. Accelerating RL with No-Reward Demonstrations

This section addresses two hypotheses: **H1**: When the unlabelled demonstrations are relevant, $\Psi\Phi$-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, $\Psi\Phi$-learning can perform at least as well as standard RL.

Figure 3 shows the results of $\Psi\Phi$-learning and the baselines in the Highway and FruitBot environments. In the single-task scenario (3a), 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, $\Psi\Phi$-learning achieves competitive results, outperforming DQN+BC-Aux (Hernandez-Leal et al., 2019; Ndousse et al., 2020). In the adversarial task (3b), 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 $\Psi\Phi$-learning does not suffer like other imitation learning methods; instead, it retains the performance of standard RL (**H2**). $\Psi\Phi$-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 (3c) is the most realistic autonomous driving task, in which other agents navigate the highway with varying driving styles. Here, $\Psi\Phi$-learning clearly outperforms all other methods, suggesting it can leverage information about other agents’ preferences in order to learn the underlying task structure of the environment, accelerating performance on the ego-agent’s RL task (**H1**). FruitBot (3d) gives consistent results, showing that $\Psi\Phi$-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 1 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 $\Psi\Phi$-learning, BC and SQIL directly learn 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**).
Table 1. We evaluate how well $\Psi\Phi$-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 DQN or PPO to train an RL agent, depending on the environment. A "$\dagger$" indicates methods that directly learn a policy from demonstrations. A "$\ddagger$" 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.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Roundabout$^{\ddagger\star}$</th>
<th>CoinGrid$^{\ddagger\star}$</th>
<th>FruitBot$^{\ddagger\star}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC* (Pomerleau, 1989)</td>
<td>$0.81 \pm 0.02$</td>
<td>$0.69 \pm 0.06$</td>
<td>$0.37 \pm 0.02$</td>
</tr>
<tr>
<td>GAIL$^{\ddagger}$ (Ho &amp; Ermon, 2016)</td>
<td>$0.77 \pm 0.07$</td>
<td>$0.73 \pm 0.02$</td>
<td>$0.31 \pm 0.02$</td>
</tr>
<tr>
<td>ITD$^{\ddagger}$ (ours, cf. Section 3.1)</td>
<td>$0.92 \pm 0.01$</td>
<td>$0.77 \pm 0.01$</td>
<td>$0.35 \pm 0.04$</td>
</tr>
</tbody>
</table>

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 4a shows the original CoinGrid game, and Figure 4b–4d shows the first three dimensions of the learned cumulant vector, $\hat{\phi}_1 \cdots \hat{\phi}_3$ (the rest are given in the Appendix). We find that $\hat{\phi}_3$ most active for red coins, $\hat{\phi}_2$ for green, and $\hat{\phi}_1$ for yellow. Clearly, ITD has learned cumulant features that span the space of goals for this game. See Appendix E for more details and visualisation of the learned rewards and preferences.

4.4. Imitation learning

Here, we investigate hypothesis H4: $\Psi\Phi$-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 $\Phi$ and the $\Psi$ and $\omega$ 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 $\Psi\Phi$-learning algorithm including ITD and learning from RL and experience to update the shared cumulants $\Phi$.

Accuracy in predicting other agents’ actions on the held-out test set is used to measure imitation learning performance. Figure 5 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, $\Psi\Phi$-learning achieves similar performance. However, when using RL to improve imitation, $\Psi\Phi$-learning performs well on both the train and test set, achieving markedly higher accuracy ($\approx 95\%$) in predicting other agents’ behaviour. This suggests that when $\Psi\Phi$-learning uses RL and interaction with the world to improve the estimation of the shared cumulants $\Phi$, this in turn improves its ability to model the $Q$.
function of other agents and predict their behaviour. Further, \( \Psi \Phi \)-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 \( \Psi \Phi \)-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**: \( \Psi \Phi \)-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 2 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 \( B^{\text{new}} \) in the new task, the \( \Psi \Phi \)-learner identifies the preference vector for the new task by solving 

\[
\sum_{t_{\text{new}}} \mathcal{L}_R(\theta_{\Psi \Phi}, w).
\]

In contrast, SQIL requires 100 episodes to reach the same performance.

### 5. 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 (Dimitrakis & 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. Our inverse TD (ITD) learning offline multi-task inverse reinforcement learning algorithm is similar to the Cascaded Supervised IRL (CSI) approach (Klein et al., 2013). However, CSI assumes a single-task, deterministic expert while ITD does not.

#### 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).
We have presented two major algorithmic contributions. Work, Lee et al. (2019) propose learning cumulants and (ii) exploration (Janz et al., 2019; Machado et al., 2020); baselines: how imitation with ITD can improve RL and that discovers salient task-agnostic environment features in recursive reasoning (Stahl, 1993; Yoshida et al., 2008) is 2019). However, we do not explore son, 1999; Lockett et al., 2007; He et al., 2016; Jaques et al., 2019). 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.

6. 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, ΨΦ-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 ΨΦ-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.

Acknowledgements. We thank Pablo Samuel Castro, Anna Harutyunyan, RAIL, OATML and IRIS lab members for their helpful feedback. We also thank the anonymous reviewers for useful comments during the review process. A.F. is funded by a J.P. Morgan PhD Fellowship and C.L. is funded by an Open Phil AI Fellowship.

References


Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovy, P., Budden, D., Cai, T., Clark, A., Danihelka, I., Fantacci, C., Godwin, J., Jones, C., Hennigan, T., Hessel, M., Kapturowski, S., Keck, T., Kemaev, I., King, M., Martens, L., Mikulik, V., Norman, T., Quan,

Table 2. 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 “\( \Phi \)” indicates methods that use a single model for all tasks, while “\( Ψ \)” 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.
ΨΦ-Learning: RL with Demonstrations using Successor Features and Inverse TD Learning


Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schuman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning


