Human-Interactive Subgoal Supervision for Efficient Inverse Reinforcement Learning

Xinlei Pan
University of California, Berkeley
Berkeley, California, USA
xinleipan@berkeley.edu

Eshed Ohn-Bar
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
eshedohnbar@gmail.com

Nicholas Rhinehart
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
nrhineha@cs.cmu.edu

Yan Xu
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
yxu2@andrew.cmu.edu

Yilin Shen
Samsung Research America
Mountain View, California, USA
yilin.shen@samsung.com

Kris M. Kitani
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
kkitani@cs.cmu.edu

ABSTRACT

Humans are able to understand and perform complex tasks by strategically structuring the tasks into incremental steps or subgoals. For a robot attempting to learn to perform a sequential task with critical subgoal states, such states can provide a natural opportunity for interaction with a human expert. This paper analyzes the benefit of incorporating a notion of subgoals into Inverse Reinforcement Learning (IRL) with a Human-In-The-Loop (HITL) framework. The learning process is interactive, with a human expert first providing input in the form of full demonstrations along with some subgoal states. These subgoal states define a set of subtasks for the learning agent to complete in order to achieve the final goal. The learning agent queries for partial demonstrations corresponding to each subtask as needed when the agent struggles with the subtask. The proposed Human Interactive IRL (HI-IRL) framework is evaluated on several discrete path-planning tasks. We demonstrate that subgoal-based interactive structuring of the learning task results in significantly more efficient learning, requiring only a fraction of the demonstration data needed for learning the underlying reward function with the baseline IRL model.

KEYWORDS

Human-in-the-loop; Inverse Reinforcement Learning; subgoals

1 INTRODUCTION

Teaching robots to perform a sequential, complex task is a long-standing research problem in robot learning. For instance, consider the task of parking a car into a narrow slot as shown in Figure 1. The autonomous vehicle may be taught to sequentially move towards the target across roads while avoiding obstacles such as other cars and white lines in the environment. One key problem that arises is that while it can be easy for the car to travel on roads, the car might struggle locating a specific turning point so that it can fit within the narrow parking slot, or struggle avoiding hitting other cars when it turns around. These issues arise because there are certain critical states, namely, subgoal states, that the agent *must* visit in order to complete the entire task. In this example, the car must turn left somewhere before it reaches the empty parking space.

Leveraging human input is one way to provide information that could be helpful for learning agents, like robots, to reach important subgoal states. Specifically, a human expert can provide demonstrations of possible trajectories to go through these critical states for the robot to follow. This type of learning, termed broadly as apprenticeship learning [1, 10], is a popular approach for leveraging human input.

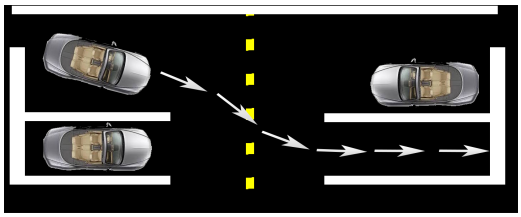

Figure 1: We develop a framework for training agents that can perform complex sequential tasks with a set of critical subgoals, such as when parking a car. In the example scenario, the car must be first positioned in a certain set of states before being able to continue and complete the goal. By interactively leveraging information regarding subgoal states and subtask demonstration as needed from a human expert, our proposed approach is shown to result in more efficient learning of the underlying reward function.

Unfortunately, expert demonstrations might not address all of the learning challenges for the following reasons: (1) ***Data Sparsity*** - While an expert can provide demonstrations of the entire task, these demonstrations are usually collected without considering the learning process (*i.e.* the structure of the task and difficulties of individual parts). Due to lack of enough demonstrations on some critical states, figuring out the way to go through them can still be difficult, which can prevent overall success. Therefore, complex sequential decision-making tasks usually require a significant amount of demonstrations to learn a reward function [15]. (2) ***Burden of Human Interaction*** - Especially in the case of human experts, constant human robot interaction is very costly and should be minimized. Unfortunately, methods like online imitation learning approaches often assume that the expert is always providing demonstrations during the entire learning process [12]. While this may be reasonable for some problems, it maybe too impractical for many other applications. (3) ***Data Redundancy*** - A full demonstration might not be needed for a learning agent equipped with

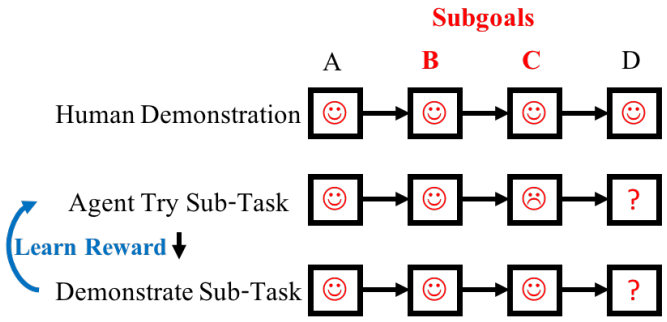

Figure 2: Diagram of our proposed approach. A human-expert can leverage subgoal states in order to efficiently interact with the learning process. Human will first provide a full demonstration covering the entire task (from A to D, these states are landmarks states in the task and there are other intermediate states not shown here), and define subgoals (B and C) and subtasks (from A to B, B to C and C to D). Then the agent will attempt the human-defined subtasks. Next, the human will only provide subtask demonstration where the agent fails. In this example, the human expert first demonstrates the entire task, and let the agent learn to perform the task. However, the agent may only finish the subtask from A to B (smiley face) but fail on the subtask from B to C (sad face), and stop at C. Then the human expert will demonstrate the subtask (B to C) that the agent failed on, and let the agent learn again. This process will repeat until the agent learns to perform the entire task.

a partial model. Given a small number of expert demonstrations, the learning agent may already know how to perform parts of the task successfully while struggling only in certain situations. In this case, it is more efficient to know where the agent fails and provide specific demonstrations for the part where the agent fails.

We make the observation that human experts can provide high-level feedback in addition to providing demonstrations for the task of Inverse Reinforcement Learning (IRL). For example, in order to teach a complex task consisting of multiple decision-making steps, a common strategy of humans is to dissect the task into several smaller and easier subtasks [9] and then convey the strategy for each of the subtasks (see Figure 2 for an example). It is reasonable that by incorporating this kind of divide-and-conquer high-level strategy coming from human's perception of the task, IRL can be more efficient by focusing on subtasks specified by human. In addition, by dividing a complex task into several subtasks using human's perception, it will be easier for humans to evaluate the performance of the current agent. Since the agent may already be able to perform well on some of the subtasks, a human expert only needs to provide feedback on subtasks that the agent struggles with.

We propose a Human-Interactive Inverse Reinforcement Learning (HI-IRL) approach that makes better use of human involvement by using *structured* interaction. Although it requires more information from the human expert in the form of subgoal states, we demonstrate that this additional information significantly reduces the required number of demonstrations needed to learn a task. Specifically, the human expert will provide critical subgoals (strategic information) the agent should achieve in order to reach the overall goal. Thus, the overall task is more "structured" and consists of a set of subtasks. We show that by using our sample efficient HI-IRL method, we can achieve expert-level performance with significantly fewer human demonstrations than several baseline IRL models. Additionally, we notice that the failure experience obtained by the agent may also be helpful to learn the reward function since the failure experience tells the agent of *what not to do*. We leverage learning from failure experience to improve reward function inference.

2 BACKGROUND

Maximum Entropy IRL. IRL typically formalizes the underlying decision-making problem as a *Markov Decision Process* (MDP). An MDP can be defined as $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $\mathcal{T}$ denotes the state transition matrix, and r is the reward function. Given an MDP, an optimal policy π^* is defined as one that maximizes the expected cumulative reward. A discount factor γ is usually considered to discount future rewards.

In IRL, the goal is to infer the reward function given expert demonstrations $\mathcal{D} = \{d_0, d_1, \cdots, d_N\}$, where each demonstration consists of state action pairs $d_i = \{s_{i0}, a_{i0}, s_{i1}, a_{i1}, \cdots, s_{ik}, a_{ik}\}$. The reward function is usually defined to be linear in the state features: $r = \theta^T \phi(s) = \theta^T f_s$, where θ is the parameter of the reward function, ϕ is a feature extractor, and f_s is the extracted state feature for state s. In maximum entropy IRL, the learner tries to match the feature expectation to that of expert demonstrations, while maximizing the entropy of the expert demonstrations. The optimization problem is defined as,

$$\theta^* = \arg\max_{\theta} - \sum_{d_i} P(d_i|\theta) \log\left(P(d_i|\theta)\right), \tag{1}$$

subject to the constraint of feature matching and being a probability distribution,

$$\sum_{d_i} P(d_i|\theta) f_{d_i} = \tilde{f}^{\mathcal{D}}, \tag{2}$$

$$\sum_{d_i} P(d_i|\theta) = 1 \text{ and } P(d_i|\theta) \geq 0, \forall i. \tag{3}$$

The expert's feature expectation can be written as

$$\tilde{f}^{\mathcal{D}} = \frac{1}{N} \sum_{d_i \in \mathcal{D}} \sum_{t=0}^{k} f_{it}. \tag{4}$$

Following current reward function θ, the policy π can be inferred via value iteration for low dimensional finite state problems. Then following π, and given initial state visitation frequency $D_{s,0} = P(S_0 = s)$ calculated from $\mathcal{D}$, the state visitation frequency at time step t can be calculated as,

$$D_{s_i,t} = \sum_{k=0}^{|\mathcal{S}|} \sum_{j=0}^{|\mathcal{A}|} D_{s_k,t-1} \pi(s_k, a_{k,j}) \mathcal{T}(s_k, a_{k,j}, s_i). \tag{5}$$

Here $\pi(s_k, a_{k,j})$ is the probability of taking action $a_{k,j}$ when the agent is at state s_k, and $\mathcal{T}(s_k, a_{k,j}, s_i)$ is the probability of transiting to state s_i when the agent is at state s_k and taking action $a_{k,j}$. The summed state visitation frequency for each state is then $D_{s_i} = \sum_t D_{s_i,t}$. The feature expectation following current policy π can be expressed as

$$f^{\pi} = \sum_{d_i} P(d_i|\theta) f_{d_i} = \sum_{s_i \in \mathcal{S}} D_{s_i} f_{s_i}. \tag{6}$$

The above optimization problem in 1 can be transformed to the following optimization problem [16],

$$\begin{aligned} \theta^* &= \arg\max_{\theta} P(\mathcal{D}|\theta) \\ &\propto \arg\max_{\theta} \exp\{ \sum_{s_i \in \mathcal{D}} \theta^T \phi(s_i)\} \\ &= \arg\max_{\theta} \exp\{ \sum_{s_i \in \mathcal{D}} \theta^T f_{s_i}\}. \end{aligned} \tag{7}$$

Optimizing Eq. 7 can be done via gradient descent on negative log-likelihood with the gradient defined by

$$\nabla_{\theta} = f^{\pi} - \tilde{f}^{\mathcal{D}}. \tag{8}$$

Maximum Entropy Deep IRL. Standard maximum entropy IRL uses a linear function to map state feature to reward value: $r = \theta^T f$. Recently, deep neural networks have demonstrated excellent performance in visual recognition and feature learning [5]. It can be beneficial to learn reward function for complex visual inputs using deep neural networks since this task may be too challenging for linear reward function. The reward function is defined as $r = g(\theta, f)$, where r is the reward value for state feature f, and θ is the neural network parameters. In the linear reward function case, the gradient of the loss function with respect to the parameters is defined as,

$$\begin{aligned} \nabla_{\theta} L &= \nabla_r L \cdot \nabla_{\theta} r \\ &= \nabla_r L \cdot f. \end{aligned} \tag{9}$$

From equation 8, we know that $\nabla_{\theta} L = f^{\pi} - \tilde{f}^{\mathcal{D}}$, which can be expressed as,

$$f^{\pi} - \tilde{f}^{\mathcal{D}} = f(D_f^{\pi} - \tilde{D}_f^{\mathcal{D}}), \tag{10}$$

where f is the feature of a particular state, D_f^{π} is the agent visitation frequency of this state, and $\tilde{D}_f^{\mathcal{D}}$ is the expert visitation frequency of this state. When deep neural network is used to represent the reward function, the gradient of the loss function with respect to the parameters can be expressed as,

$$\begin{aligned} \nabla_{\theta} L &= \nabla_r L \cdot \nabla_{\theta} r \\ &= \nabla_r L \cdot \nabla_{\theta} g \\ &= (D_f^{\pi} - \tilde{D}_f^{\mathcal{D}}) \cdot \nabla_{\theta} g. \end{aligned} \tag{11}$$

IRL from Failure. While maximum entropy IRL tries to match the expected feature counts of the agent's trajectory with the feature counts of expert demonstration, it is reasonable to keep the expected feature counts following current learned reward different from that of failure experience. The learning from failure algorithm proposed in [13] demonstrates the possibility of incorporating failure experience to improve IRL. Given both successful demonstrations $\mathcal{D}$ and failure experience $\mathcal{F}$, we define linear reward function parameter θ_d and θ_f for reward function learned from $\mathcal{D}$ and $\mathcal{F}$ respectively. The goal is to maximize the probability of successful demonstrations, and match the feature expectation of successful demonstrations, while maximizing the feature expectation difference with failure experiences. In [13], the optimization problem is defined as following,

$$\begin{aligned} \max_{\pi, w, z} \quad & H(\mathcal{D}) + wz - \frac{\lambda}{2}\|w\|^2, \\ \text{subject to:} \quad & \tilde{f}^{\mathcal{D}} = f^{\pi}, \\ & f^{\pi} - \tilde{f}^{\mathcal{F}} = z, \\ & \sum_a \pi(s,a) = 1 \quad \forall s \in \mathcal{S}, \\ & \pi(s,a) \geq 0 \quad \forall a \in \mathcal{A}, \end{aligned} \tag{12}$$

where $H(\mathcal{D})$ is the causal entropy of the successful demonstrations $\mathcal{D}$, and is defined as,

$$H(\mathcal{D}) = -\sum_t \sum_{s_{1:t} \in \mathcal{S}, a_{1:t} \in \mathcal{A}} P(a_{1:t}, s_{1:t}) \log(P(a_t|s_t)), \tag{13}$$

where $P(a_t|s_t) = \pi(s_t, a_t)$ is the policy, and

$$P(a_{1:t}, s_{1:t}) = P(s_{1:t-1}, a_{1:t-1}) \mathcal{T}(s_{t-1}, a_{t-1}, s_t) \pi(s_t, a_t) \tag{14}$$

is the probability of trajectory from time 1 to time t. In Eq. 12, w is the Lagrange multiplier of z, which is a variable representing the difference between the feature expectation of failure experiences and the feature expectation following current policy π. The Lagrangian of Eq. 12 gives the following loss function,

$$\begin{aligned} \mathcal{L}(\pi, w, z, \theta_d, \theta_f) =& H(\mathcal{D}) + wz - \frac{\lambda}{2}\|w\|^2 \\ &+ \theta_d(f^{\pi} - \tilde{f}^{\mathcal{D}}) \\ &+ \theta_f(f^{\pi} - \tilde{f}^{\mathcal{F}} - z). \end{aligned} \tag{15}$$

Following the optimization in [13], the optimization step update for θ_d and θ_f is,

$$\begin{aligned} \theta_d &= \theta_d - \alpha(f^{\pi} - \tilde{f}^{\mathcal{D}}), \\ \theta_f &= \frac{(f^{\pi} - \tilde{f}^{\mathcal{F}})}{\lambda}, \end{aligned} \tag{16}$$

where α is the learning rate for θ_d and λ is a learning rate for θ_f which is annealed throughout the learning. More details of the learning from failure approach can be found in [13]. The description of the IRL from failure approach is described in Algorithm 1.

Algorithm 1 Deep IRL from Failure (IRLFF)

Require: Failure experience $\mathcal{F}$, expert demonstration $\mathcal{D}$
Require: State Transition Matrix $\mathcal{T}, \alpha, \alpha_\lambda, \lambda, \theta_d^t, \lambda_{min}$, all feature input f, where θ_d^t is a deep neural network
Return: Updated reward function θ_d, θ_f
Start:
$\tilde{f}^{\mathcal{D}}$ = FeatureCount($\mathcal{D}$) *(Eq. 4 with $\mathcal{D} = \mathcal{D}$)*
$\tilde{f}^{\mathcal{F}}$ = FeatureCount($\mathcal{F}$) *(Eq. 4 with $\mathcal{D} = \mathcal{F}$)*
$P_0^{\mathcal{D}}$= initialStateDistribution($\mathcal{D}$)
$P_0^{\mathcal{F}}$ = initialStateDistribution($\mathcal{F}$)
$\theta_f = \mathbf{0}$
$\theta_d = \theta_d^t$
Repeat:
$r = g(\theta_d, f) + \theta_f \cdot g_{fc1}(\theta_d, f)$
π = SoftValueIteration(r)
$f_{\mathcal{F}}^{\pi}$ = FeatureExpectation($P_0^{\mathcal{F}}, \pi, \mathcal{T}$)
$f_{\mathcal{D}}^{\pi}$ = FeatureExpectation($P_0^{\mathcal{D}}, \pi, \mathcal{T}$)
$\theta_d = \theta_d - \alpha(D_f^{\pi} - \tilde{D}_f^{\mathcal{D}}) \cdot \nabla_{\theta_d} g$
θ_f calculated according to Eq. 19
if $\lambda > \lambda_{min}$:
$\lambda = \alpha_\lambda \lambda$
until convergence

3 HUMAN-INTERACTIVE INVERSE REINFORCEMENT LEARNING (HI-IRL)

We propose Human-Interactive Inverse Reinforcement Learning (HI-IRL) to make more efficient use of human participation beyond simply providing demonstrations. Different from approaches such as [16], we require more human-agent interactions during the learning process by allowing the agent try out subtasks defined by a human and letting the human provide further demonstrations on subtasks if the agent struggles (we provide formal definition of "struggle" later in this section). Different from approaches such as DAGGER [12], humans do not need to constantly provide entire demonstrations; instead demonstrations are obtained only when required by the agent. There indeed can be other forms of human interaction when the agent struggles, some of which are compared to as baselines in the experiments. For example, the human may continue to provide the entire demonstrations when the agent struggles, similar to the approach in [12]. However, we find this method of interaction to be less effective. A second possibility is to simply let the agent try the same task repeatedly, until it happens to finish the task. Then, the successful trajectory that the agent experienced can be used as human demonstration. However, this approach is limited in scenarios with large state spaces. In addition to being highly inefficient, even if the agent reaches the goal, the trajectory that the agent traveled may not be an optimal or a desired trajectory. In contrary, we show that our method of structuring the interaction enables better efficiency on complex tasks. Next, we first describe our method, HI-IRL, and then give a demonstration of the optimality of our subgoal selection strategy.

3.1 HI-IRL

Step 1: Human expert provides several full demonstrations and define subgoals. Given a task consisting of multiple decision making steps, the human expert will first provide N full demonstrations $\mathcal{D} = \{d_0, \cdots, d_N\}$ completing the entire task. The number of demonstrations in $\mathcal{D}$ can be relatively small, for example, 1 or 2 demonstrations to learn an initial reward function. The human expert will then dissect the entire task into several parts by indicating critical subgoal states where the agent must go through in order to achieve the overall task. For example, in an indoor navigation task, the agent tries to find a way from one room to anther, the state when the agent is at the exit between the two rooms is a critical subgoal state. While trajectories with different starting position in the first room and different goal position in the second room varies, they all need to go through the critical state corresponding to the exit.

We denote these critical subgoal states as $\mathcal{S}_{sub}$. One typical characteristics of these subgoal states is that the probability of any expert trajectories to include them will be close to 1,

$$P(s_i \in d_j) \approx 1, \forall s_i \in \mathcal{S}_{sub} \text{ and } \forall d_j \in \mathcal{D}. \tag{17}$$

The reason why it may not be 1 is to allow cases where there are multiple states functioning very similar as subgoal states. For instance, there are multiple exits from one room to another in the indoor navigation example. In this case, the probability of any expert trajectories to include any one of these states will be 1.

Given these subgoal states $\mathcal{S}_{sub}$, any trajectory $\xi = \{s_0, \cdots, s_k\}$ can be dissected into several subtasks $T_{sub} = \{d_1, d_2, \cdots, d_m\}$, where m is the number of subtasks within this trajectory ξ , and concatenating these subtasks together will get the original trajectory ξ. The starting state and end state of each of these subtasks except d_1 and d_m belong to $\mathcal{S}_{sub}$. The end state and starting state of d_1 and d_m, respectively, belong to $\mathcal{S}_{sub}$. A more formal definition of trajectory dissection is to consider all possible trajectories from a chosen start state to goal state as a set $\Xi = \{\xi_1, \cdots, \xi_x\}$, and subgoal states are defined by,

$$\mathcal{S}_{sub} = \bigcap_{i=1}^{x} \xi_i. \tag{18}$$

Step 2: Agents tries the defined subtasks. Starting from a randomly selected starting state s_r, the agent will be required to reach each of the subgoals sequentially towards the ultimate state s_{goal}. This means that given the optimal path from the agent's current state s_r to the goal state s_{goal}: $\xi_{s_r \to s_{goal}} = \{s_r, \cdots, s_{sub1}, \cdots, s_{sub2}, \cdots, s_{subk}, \cdots, s_{goal}\}$ where the agent is expected to reach subgoal states along the path from s_{sub1} to s_{subk} sequentially. If the agent successfully arrives to subgoal s_{subi} within $step_{min, s_{subi}} + step_{thr}$, the agent will be required to reach the next subgoal $s_{sub(i+1)}$ starting from current state s_{subi}. Here, $step_{min, s_{subi}}$ is the minimum steps required to reach s_{subi} from the start state s_r, and $step_{thr}$ is the extra threshold steps to allow some exploration.

Step 3: Human provides further demonstrations if needed. Depending on the performance of the agent on the subtasks, if the agent successfully finished all subtasks, then the human expert will not provide further demonstrations. The human expert will only provide demonstrations on subtasks that the agent struggles.

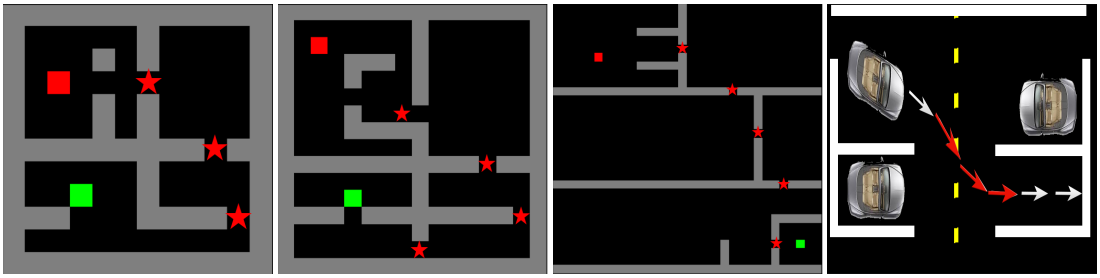

Figure 3: Subgoals specified in 12x12, 16x16, 32x32 grid world environment, and car parking environment. In the grid world environment, states are defined as the grid position the agent is current at (specified by red box), goals are represented by green box, and subgoals are indicated by red stars. In the car parking environment, states are defined as the car global coordinate as well as the orientation of the car, which can be represented by an arrow. The subgoals in the car parking environment is specified by a set of red arrows.

Algorithm 2 Human-Interactive Inverse Reinforcement Learning (HI-IRL)

Require: Set of initial demonstrations d_0, T, State Transition Matrix $\mathcal{T}$, θ^0, all state raw feature f, and human $\mathcal{H}$.
Return: Reward function θ^{T+1}
Define: $\mathcal{D}$: positive demonstrations; $\mathcal{F}$: failure experience; $\mathcal{E}$: agent experience; $\mathcal{S}_{sub}$: set of subgoal states
Start:
$\mathcal{S}_{sub}$ = specify_subgoals($\mathcal{H}$)
$\mathcal{D} = d_0$;
θ^1 = MaxEntIRL($\mathcal{D}$, θ^0)
for t $\in 1, 2, ..., T$
$\mathcal{E}$ = Rollout(θ^t, $\mathcal{S}_{sub}$)
for e in $\mathcal{E}$
$\mathcal{F}, \mathcal{D}$ = UpdateDemo(e, θ^t, $\mathcal{D}$)
$\theta_d^{t+1}, \theta_f^{t+1}$ = IRLFF($\mathcal{F}, \mathcal{D}, \mathcal{T}, \theta_d^t, f$) *(Alg. 1)*
$\theta^{t+1} = (\theta_d^{t+1}, \theta_f^{t+1})$

For example, if the agent is not able to complete a subtask ending in subgoal s_{subi}, then human will provide further demonstrations on this subtask. Since these additional demonstrations may not be complete demonstrations starting from the very beginning state to the ultimate goal state, we refer to these demonstrations as *partial demonstrations*. The initial demonstrations mentioned in step 1 are referred as *full demonstrations*. This intuitive interaction scenario is formally defined below.

Suppose the agent is given a subtask to go from state s_i to state s_j. The minimum number of steps to travel from s_i to s_j is $step_{min,s_i \to s_j}$, and to allow some level of exploration, the agent will be given extra $step_{thr}$ steps to reach s_j. The value $step_{thr}$ depends on the difficulty of specific task, if the task is fairly difficult, we set it to a high value, otherwise, we set it to a low value. In our approach this value can be regarded as a hyper-parameter that needs to be tuned. Struggling is defined as the scenario where the agent is not able to reach s_j within $step_{thr} + step_{min,s_i \to s_j}$. Here, the human will provide further demonstrations on this particular task (from s_i to s_j).

Step 4: Learning reward function from both failure experiences and expert demonstrations. When the agent fails to finish some subtasks, it gains failure experiences, denoted as $\mathcal{F}$. These demonstrations are not given by human, but instead by the learning agent itself. The expert's further demonstrations are denoted as $\mathcal{D}$, which already includes the initial full demonstrations. Since learning from failure approaches [13] generally focus on the linear reward function case, we propose to use a deep neural network to extract features from raw states, and then use a linear reward function to get reward value from these extracted features.

Our deep neural network reward function takes in input in the form of raw states (*i.e.,* images) and process it with three convolutional layers with each one followed by batch normalization layers and ReLU activation. Two fully connected layers are followed to output the final reward value. The last layer outputs a scalar value which will be used as the reward value corresponding to θ_d in Eq. 16. The second last layer output vector will be used to calculate θ_f in Eq. 16. If we denote the network parameters as $\theta_d = \{conv, bn, ReLU, FC_1, FC_2\}$, the network input as f, and the network function as $r_d = g(\theta_d, f)$, then we have

$$FC_{1,out} = g(conv, bn, ReLU, FC_1, f) \doteq g_{fc1}(\theta_d, f)$$
$$\theta_f = \frac{FC^{\pi}_{1,out} - \widetilde{FC}^{\mathcal{F}}_{1,out}}{\lambda} \tag{19}$$

Here θ_d will be the neural network and θ_f will be a vector of the same size as $FC_{1,out}$, $FC^{\pi}_{1,out}$ is the feature expectation following the current policy π, and $\widetilde{FC}^{\mathcal{F}}_{1,out}$ is the feature expectation of failure experience $\mathcal{F}$. The final reward function will be $r = g(\theta_d, f) + \theta_f \cdot g_{fc1}(\theta_d, f)$. The detailed learning from both failure experience and expert demonstration algorithm is described in Algorithm 2.

3.2 Optimality of Subgoal Selection

In HI-IRL, the human will specify critical subgoal states $\mathcal{S}_{sub}$ which have a very high probability to be included in any expert demonstrations, and other non-critical states will have relatively lower probability to be included in any expert demonstrations. Define $\mathcal{S}_{nc} \doteq \mathcal{S} \setminus \mathcal{S}_{sub}$ as all states except human defined subgoal states. Given two trajectories $\xi_1 = \{s_{1,0}, s_{1,1}, \cdots, s_{1,k}\}$ and

$\xi_2 = \{s_{2,0}, s_{2,1}, \cdots, s_{2,k}\}$, where $s_{1,i} = s_{2,i}, \forall i \in \{0, \cdots, k-1\}$, and $s_{1,k} \in \mathcal{S}_{sub}$ and $s_{2,k} \in \mathcal{S}_{nc}$, intuitively, ξ_1 will be favored over ξ_2,

$$\begin{aligned} & P(\xi_1) > P(\xi_2) \\ \Rightarrow \exp \sum_{i=1}^{k} r(s_{1,i}) & > \exp \sum_{i=1}^{k} r(s_{2,i}), \\ \Rightarrow r(s_{1,k}) & > r(s_{2,k}), \end{aligned} \tag{20}$$

which means that critical subgoal states will have higher reward than non-subgoal states around them. In the linear reward function case, the reward function parameter θ is optimized when,

$$\tilde{f}^{\mathcal{D}} = \sum_{i=1}^{|\mathcal{S}|} D_{s_i} f_{s_i}, \tag{21}$$

which means the final policy will favor states that appear more times in expert demonstrations $\mathcal{D}$ in order to match the feature expectation of $\mathcal{D}$. Given two states s_1 and s_2, and define $p(s_1, \mathcal{D})$ as the frequency of s_1 appears in $\mathcal{D}$, the same for s_2, and suppose $p(s_1, \mathcal{D}) > p(s_2, \mathcal{D})$, then we have,

$$\begin{aligned} & D_{s_1} > D_{s_2} \\ \Rightarrow P(\xi_1 | s_1 \in \xi_1) & > P(\xi_2 | s_2 \in \xi_2), \end{aligned} \tag{22}$$

where ξ_1 and ξ_2 are two trajectories, where all other states are same, except that ξ_1 contains s_1 while ξ_2 contains s_2. Given Eq. 20, we know that $r(s_1) > r(s_2)$, which means states that appear more times in expert demonstrations will typically have higher rewards. Therefore, in order to make sure those critical states have higher rewards, we must increase the demonstrations around them. By letting human specify these critical states, and providing extra demonstrations if the agent struggles, we ensure that these states receive more attention during demonstration collection, which leads to better reward function learning.

4 EXPERIMENTS

We designed the experiment parts to demonstrate the key contributions of our proposed HI-IRL method. **First**, we demonstrate that by leveraging human interaction in inverse reinforcement learning, we obtain better data efficiency than traditional inverse reinforcement learning approach that trains on offline collected data (the standard maximum entropy IRL method). **Second**, we provide a better human interaction strategy where the burden on human can be reduced compared with existing methods such as [12]. **Third**, we demonstrate that by carefully selecting the key subgoals, it achieves better reward function learning than random selection of subgoals. The experimental environments are designed to be complex sequential decision making process with critical subgoal states that the agent must go through in order to complete the overall task.

Baselines. In order to show the key contributions of our HI-IRL method, we compare our algorithm with (1) maximum entropy IRL (here after denoted as **`MaxEntIRL`**); (2) human interactive IRL without specifying subgoals (here after denoted as **`HI-IRLwos`**), which is similar to approach like [12]; and (3) human interactive IRL with randomly selected subgoals (here after denoted as **`HI-IRLwr`**). In human interactive IRL without specifying subgoals, the procedure is similar to our method, except that the agent will be required to complete entire task and human expert will provide full demonstrations if the agent struggles. The purpose of comparing with **`MaxEntIRL`** is to show the benefits of interacting with human during the learning process (our **first** contribution). While both **`HI-IRLwos`** and **`HI-IRLwr`** have human interaction, **`HI-IRLwos`** tries to provide the entire demonstration again which contains redundancy and increases human burden; **`HI-IRLwr`** tries to provide demonstrations for randomly selected subtasks, which fails to emphasize on critical subgoal states, and may lead to ill reward function learning. The purpose of comparing with **`HI-IRLwos`** is to show the benefits of subgoal selection as it reduces human burden to demonstrate entire task (our **second** contribution). The purpose of comparing with **`HI-IRLwr`** is to show the benefits of selecting critical subgoals instead of random subgoals (our **third** contribution).

We performed several sets of experiments in grid-world and car parking environments spanning different scales of state space. All environments contain critical subgoal states that the agent *must* go through to complete the entire task. In all experiments, we use deep neural network to represent reward function. In the grid-world environment, the network is composed of three layers of convolutional neural network with each followed by a batch normalization layer and ReLU activation layer, then two fully connected layers are followed to output the final reward value. In the car parking environment, the network is similar to the network in grid-world environment, except there are 2 convolutional layers due to smaller input image size.

Grid-world Environment. The grid-world environment involves grid-world navigation where the agent is put in a place at the beginning and the task is to find a way to a target position. In this experiment, grid-world of different scales of state space are used for evaluation. Specifically, a 12x12, a 16x16, and a 32x32 grid-world environment are used. Regions in the grid-world where there are obstacles are not counted towards agent state. The state space in this game is the grid world image, and the action space consists of 5 actions, stay in place, go up, go down, go left and go right.

Since all four methods require some initial human demonstration to learn a reward function, a certain number of human demonstrations $\mathcal{D}$ are collected at the beginning. In both the gridworld environment and car parking environment, we have finite number of states and the optimal path from one state to another can be automatically solved by using the Dijkstra algorithm [14]. Therefore, we generate the demonstration automatically instead of getting them from real human. However, human expert will specify critical subgoal states $\mathcal{S}_{sub}$ to be used in our method. A set of test starting state will be specified by human that is different from the training data $\mathcal{D}$. Then $\mathcal{D}$ is used to get the reward function following **`MaxEntIRL`** method. One demonstration randomly sampled from $\mathcal{D}$ will be used for training initial reward function for our method, **`HI-IRLwos`** method, and **`HI-IRLwr`** method. In **`HI-IRLwos`**, the agent will be required to start from a randomly selected starting state, and find a way to the final target state, and human will provide further demonstration if the agent struggles. In **`HI-IRLwr`**, randomly selected subgoals will be used to define subtasks, and the agent will try to complete these subtasks, and human will provide further demonstrations if needed. All four methods are trained with

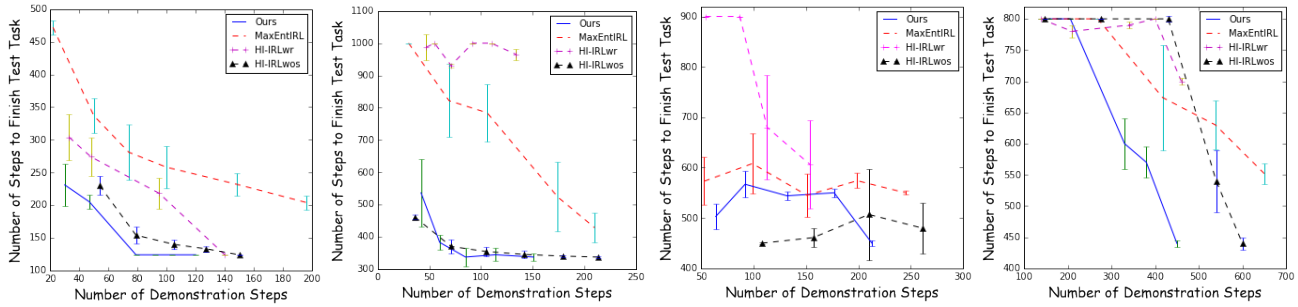

Figure 4: Number of demonstration steps VS number of steps used to complete the same test tasks curve. (a): 12x12 Grid-world; (b) 16x16 Grid-world; (c) 32x32 Grid-world. (d) Car parking environment.

the same learning rate and number of iterations. Different number of demonstrations are used to train reward function and then evaluate on the same test task 5 times to get the mean value of test performance.

Car-Parking Environment. Parking a car into a garage spot involves driving the car to a place near the slot, adjust the orientation of the car and drive the car into the parking box without hitting obstacles. In this environment, it is critical that the car has to stop at a certain state near the parking slot to ensure that after adjusting the orientation, the car will not hit obstacles. The car parking environment interface is shown in Figure 1. The number of agent possible states is about 5k – much larger than the state space in the grid-world environment. The state of this game is the car parking interface, and the action space consists of stay in place, move forward, move back, rotate to the left and rotate to the right.

At the beginning, human demonstrations and human specified subgoals are collected. Then follow the same procedure as in the grid-world environment, we obtained training results for all four methods. The subgoals selected for each environment is visualized in Figure 3.

4.1 Results and Analysis

Grid-world Environment. The number of demonstration steps versus number of steps used to complete the same test tasks curve is shown in Figure 4, which includes the results for all four methods. The test task is to set the agent at some initial states on the top left region in the grid world, and then require the agent to travel to the same destination as in training time. Since the goal of our approach is to reduce the burden of human, for example, the human will provide less demonstrations, the results indicate that our method achieves better human interaction efficiency and the agent learns to complete the same test task with less but more informative demonstration from human. The reason why the **`MaxEntIRL`** method works worse than the other three methods is that there are much more training data to learn from in this method. Therefore, it may require more iterations to train, which is another burden of this method. The **`HI-IRLwr`** method works in the 12-by-12 state size case, but does not work in the 16-by-16 state size case. The reason is that the subgoals are randomly selected, which means there is a probability that they are selected to be near the critical subgoal states, achieving similar performance as our method. Our method uses slightly more steps to complete the test task in the 32-by-32 grid-world at initial training than **`HI-IRLwos`** method. However, as indicated in the figure, we can use less steps of demonstrations but achieve similar performance.

Car-Parking Results. The car-parking results include the number of demonstrations versus number of steps to complete the same test tasks curve shown in Figure 4. Our method achieves near oracle performance with less demonstrations from human than other baselines. Since this MDP contains much richer states (in total 5k states) than previous MDPs, this experiment demonstrates that our method has the abilility to generalize to large state space case.

5 RELATED WORK

Inverse Reinforcement Learning (IRL). IRL is a method that infers a reward function given a set of expert demonstrations [1, 10]. One of the key assumptions of IRL is that the observed behavior is optimal (maximizes the sum of rewards). Maximum entropy inverse reinforcement learning [16] employs the principle of maximum entropy to learn a reward function that maximizes the posterior probability of expert trajectories. Though [16] relaxes the optimality constraints, it cannot handle significantly suboptimal demonstrations. [16] also does not consider the redundancy of demonstrations. In our case, since we have both agent's failure experience as defined later and expert's demonstrations, we can leverage the failure experience to improve the current reward. By using human feedback interactively in the training, our method aims to ultimately improve the reward inference process. By interacting with the human only when needed, we are also able to reduce the amount of human involvement (*i.e.,* redundant demonstration data).

Human-in-the-Loop IRL. Leveraging different types of human input during training has been previously shown to improve performance accuracy and learning efficiency. In [3], the human and robot collaborate with each other to maximize the human's reward. Yet, [3] assumes that the underlying reward function for every state is visible for the human, which may not be practical for many RL problems. One reason for this is that the human usually knows what action to take under a specific state, but it is hard to infer the value function of states as it triggers another IRL problem. In [11], agents constantly seek advice from a human for clustered

states, and so the learned reward gradually improves. However, creating the state clusters and give general advice for particular clusters is itself a demanding task for the human, since the states within a cluster may not have the same optimal policy and the human has to tradeoff to make a decision. The work of [2] studied the safety of AI by giving human feedback when the agent is performing sub-optimally, the method can reduce the amount of human involvement to learn a safe policy. However, the problem studied is different from ours since we focus on improving IRL performance on complex sequential decision-making tasks instead of AI safety. As a human-in-the-loop imitation learning algorithm, DAGGER [12] has proven to be effective in reducing the covariate shift problem in imitation learning. However, [12] does not explicitly learns a reward function and requires constant online interaction.

Hierarchical IRL. Hierarchical reinforcement learning [6] was proved to be effective in learning to perform challenging tasks with sparse feedback by learning to optimize different levels of temporal reward functions. Hierarchical IRL [4] was recently proposed to learn the reward function for complex tasks with delayed feedback. The work of [4] shows that by segmenting complex tasks into a sequence of subtasks with shorter horizons, it is possible to obtain optimal policy more efficiently. However, since [4] does not get expert feedback during learning, and does not explicitly leverages partial demonstrations, it may still involve redundant demonstrations.

Learning from Failure. Traditional IRL assumes the demonstrations by experts are optimal in the sense that it optimizes the sum of reward [8, 10, 16]. Recently, learning from failure experience has been proven to be beneficial with properly defined objective functions [7, 13]. Inspired by [13], we complement the human-in-the-loop training process with learning from failure experience experienced by agents, as we find it to improve reward function inference.

6 CONCLUSIONS AND REMARKS

Motivated by the need to address challenges when learning complex sequential decision-making with an IRL framework, this paper presents a framework for leveraging structured interaction from a human during training. In addition to providing demonstrations of the task to be performed by a learned agent, the method also leverages the human's high level perception about the task (in the form of subgoals) in order to improve learning. Specifically, humans can transfer their divide-and-conquer approach for problem solving to inverse reinforcement learning by providing segmentation of the current task and a set of subtasks. Additional improvements are made by employing the agent's own failure experience in addition to the human's demonstrations. Experiments on a discrete grid-world path-planning task and large state space car parking environment demonstrated how subgoal supervision resulted in more efficient learning.

For future work, we would like to apply HI-IRL for additional tasks with increasing complexity. Incorporating HI-IRL with a real-world robot experiment could further support its use in applications where input from a human is helpful but costly to acquire. In addition, it is also interesting to explore automatic optimal task dissection to further reduce human burden.

REFERENCES

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

[2] K. Amin, N. Jiang, and S. Singh. 2017. Repeated Inverse Reinforcement Learning. *ArXiv e-prints* (May 2017). arXiv:cs.AI/1705.05427

[3] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. 2016. Cooperative inverse reinforcement learning. In *Advances in Neural Information Processing Systems*. 3909–3917.

[4] Sanjay Krishnan, Animesh Garg, Richard Liaw, Lauren Miller, Florian T. Pokorny, and Ken Goldberg. 2016. HIRL: Hierarchical Inverse Reinforcement Learning for Long-Horizon Tasks with Delayed Rewards. *CoRR* abs/1604.06508 (2016). http://arxiv.org/abs/1604.06508

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[6] Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. 2016. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. *CoRR* abs/1604.06057 (2016). http://arxiv.org/abs/1604.06057

[7] Kyungjae Lee, Sungjoon Choi, and Songhwai Oh. 2016. Inverse reinforcement learning with leveraged Gaussian processes. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 3907–3912.

[8] Sergey Levine, Zoran Popovic, and Vladlen Koltun. 2011. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems*. 19–27.

[9] Sanmit Narvekar, Jivko Sinapov, Matteo Leonetti, and Peter Stone. 2016. Source task creation for curriculum learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 566–574.

[10] Andrew Y Ng, Stuart J Russell, et al. 2000. Algorithms for inverse reinforcement learning.. In *Icml*. 663–670.

[11] Phillip Odom and Sriraam Natarajan. 2016. Active advice seeking for inverse reinforcement learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 512–520.

[12] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. 2011. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[13] Kyriacos Shiarlis, Joao Messias, and Shimon Whiteson. 2016. Inverse reinforcement learning from failure. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1060–1068.

[14] S Skiena. 1990. DijkstraâĂŹs algorithm. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley* (1990), 225–227.

[15] Markus Wulfmeier, Dushyant Rao, and Ingmar Posner. 2016. Incorporating Human Domain Knowledge into Large Scale Cost Function Learning. *arXiv preprint arXiv:1612.04318* (2016).

[16] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum Entropy Inverse Reinforcement Learning.. In *AAAI*, Vol. 8. Chicago, IL, USA, 1433–1438.