Learning reward hierarchy with safe exploration for Inverse Reinforcement Learning

Naresh Balaji nbrav@kth.se
Anna Hedstrom annaheds@kth.se
Yonk Shi pyshi@kth.se

INTRODUCTION

For reinforcement learning (RL) systems in complex environments, one of the the most challenging tasks is to design an appropriate reward function. In many scenarios, there are no ways to access the *true* reward function and similarly it may be hard to ensure safe behaviour while exploring the state space. Further there can be intrinsic hierarchies to goals. In order to tackle this problem, we combine a modified Maximum Entropy algorithm [4]with hierarchical elements to safely learn reward functions from expert demonstrations.

PROBLEM FORMULATION

The objective is to recover a learned reward hierarchy, specified as $\mathcal{R}(\tau)$, from sampled expert trajectories by learning parameters ψ. For this we consider the class of subtasks that can be defined by a finite-horizon Markov Decision Process $\mathcal{MDP}$:

$$\langle \mathcal{S}, \mathcal{A}, \gamma, \mathcal{T}, \mathcal{R} \rangle$$

where $\mathcal{S}$ is a finite set of states, $\mathcal{A} = \{a_1, ...a_k\}$ is a finite set of actions, $\gamma \in$ [0,1) is the discount factor for future rewards, $\mathcal{T}$ is the transition model that is the probability of reaching state $s\prime$ given that action a is taken in state s i.e., $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, and $\mathcal{R}$ is a reward function that is a sequence of sub-reward functions, $\mathcal{R} = [\mathcal{R}_1, ..., \mathcal{R}_k]$ where $\mathcal{R}_i : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$.

For each $\mathcal{R}_i$ we associate a sub-goal $\rho_i \subseteq \mathcal{S}$, representing tasks $\mathcal{G} = [\rho_1, ..., \rho_k]$. Once all the $\rho_i \in \mathcal{G}$ are reached sequentially, the policy π that maps states to actions will be deemed successful. Furthermore, the policy is *optimal* when it is successful.

Let $\mathcal{T}$ be a set of demonstrations denoted as $\mathcal{T} = \{\tau_1, ..., \tau_T\}$ of a sequential task. The reward function can be recovered from trajectories τ, specified as $\mathcal{R}(\tau)$.

Since the expert follows an optimal policy $\pi^\star$, it is plausible to assume that the expert trajectories are likely to lead to high utility behaviour in $\mathcal{MDP}$ environments for which it was designed. It is however hard to verify that the expert actually captured the *true* reward function. In cases of just slightly changing $\mathcal{MDP}$, e.g., new terrain types introduced, we can no longer guarantee that the inverse agent will adopt the right behaviour that the expert intended. For example, if our expert do not consider that the inverse agent might encounter risky terrain types such as "lava" in the system and thus leaves that out of the reward specific, optimising the expert's reward may lead to dangerous behaviour [2]. We will therefore distinguish between the true reward function $\mathcal{R}_{true}$ and the intended proxy reward function $\mathcal{R}_{proxy}$ (only known to the expert).

Since no uncertainty is taken into account in the original IRL formulation, inevitably, when the inverse agent begins to interact with the test $\mathcal{MDP}$, safe exploration cannot be asserted. To mitigate unintended consequences from miss-specified reward functions, we introduce a *safety criterion* for determining uncertainty level for each state and *risk-averse planning* for cases as such.

The $\mathcal{R}_{true}$ encodes that the agent should first capture the flag and then reach the goal, while only traversing road and avoiding lava and the $\mathcal{R}_{proxy}$ function is not aware of lava and therefore does not include that such states has to be avoided. To this methods on safe exploration, we also investigate work with respect feudal networks.

METHODS

One observation that can be made for hierarchical tasks is that states that are goals and sub-goals are more likely to occur in the expert trajectories than all other states [3]. Correspondingly, the entropy of those states will be lower across the trajectories. For this reason, we, along with maximizing the log-likelihood of the reward-function, minimize the entropy of the states in the trajectories.

We first start by defining a reward function that is the softmax of linear function of features (unlike the linear function in MaxEnt). This makes sure that the reward function stays as a probability distribution and allows easier entropy gradient computation.

$$r(s; \boldsymbol{\psi}) = \text{softmax } \boldsymbol{\psi}^T \phi(s)$$

The entropy of the reward function can be defined as $H(r(\circ; \psi)) = \sum_s r(s; \boldsymbol{\psi}) \log r(s; \boldsymbol{\psi})$. The gradient of the reward function can be derived to be the following:

$$\nabla_{\boldsymbol{\psi}} H(r(\circ; \boldsymbol{\psi})) = \sum_s r(s; \boldsymbol{\psi}) \boldsymbol{\phi}(s) \big(\log r(s; \boldsymbol{\psi}) - H(r(\circ; \boldsymbol{\psi})) \big)$$

We can now add the entropy term to the maximal-likelihood function, and the resulting gradient can be shown to be:

$$\nabla_{\boldsymbol{\psi}} L(r(\circ; \boldsymbol{\psi})) = \frac{1}{M} \sum_{\tau_d \in D} \sum_{s_d^t \in \tau_d} r(s_d^t; \boldsymbol{\psi}) \, (\boldsymbol{\phi}(s_d^t) -$$

$$\sum_{s' \in S} r(s'; \boldsymbol{\psi}) \boldsymbol{\phi}(s')) - \sum_s p(s|\boldsymbol{\psi}) \, r(s; \boldsymbol{\psi}) \, (\boldsymbol{\phi}(s) -$$

$$\sum_{s'} r(s'; \boldsymbol{\psi}) \boldsymbol{\phi}(s')) - \sum_s r(s; \boldsymbol{\psi}) \boldsymbol{\phi}(s) \big(\log r(s; \boldsymbol{\psi}) - H(r(\circ; \boldsymbol{\psi})) \big)$$

Our notion of safety is concerned with transitions to states s that lead to fatal outcomes. Let S_k be a set of known terrain types and S_u the set of unknown terrain types.

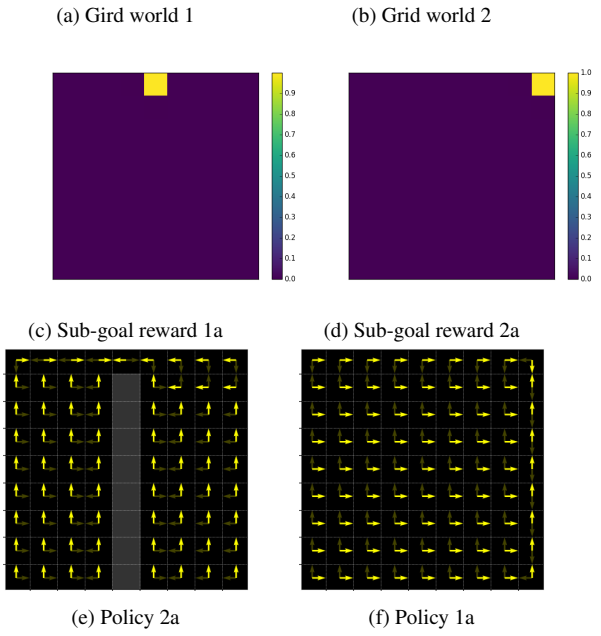
1. Safety criterion. When the inverse agent receives a new expert trajectory $\tau = (\tau_1, ...\tau_T)$ it checks whether there are unknown states in the $\mathcal{MDP}s$, given access to feature indicators $\phi(s)$ for both training and test $\mathcal{MDP}$.
2. Risk-averse planning. If risk zones s_u was identified, the agent then retrieves the nearest neighbours of such states and thereafter classifies the actions of entering such a risk zones as forbidden, by setting the $\{\mathcal{P}^a_{ss\prime}\} = 0$ for each value iteration phase in the MaxEnt algorithm.

The inverse agent should learn to distinguish the cases of *trust* (treat expert trajectories as fixed) and *distrust* (divert from policy in case of uncertainty).

EXPERIMENTAL RESULTS

We evaluated our approaches in a object gridworld, using a modified interface of Minimalistic Gridworld Environment for OpenAI Gym [1]. The agents move in the four cardinal directions $\langle up, down, left, right \rangle$. The terrain types are $\langle goal, flag, wall, road, lava \rangle$.

The results show that combining a safety criterion with risk-averse planning allowed the agent to avoid unforeseen scenarios. The major drawback of this risk-averse planning is that it forces the agent to also avoid potentially good states. However this safety threshold can be adjusted based on risk-appetite.



(a) Gird world 1 (b) Grid world 2

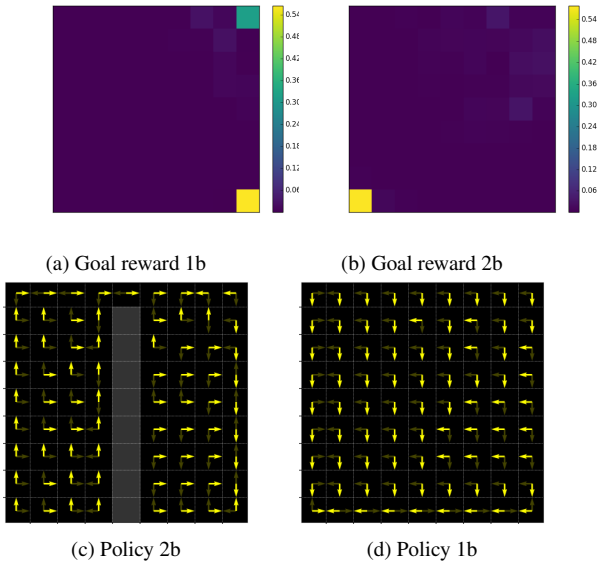
(c) Sub-goal reward 1a (d) Sub-goal reward 2a

(e) Policy 2a (f) Policy 1a

Figure 1: My complicated figure

Feudal Networks

The original feudal network consisted of a ConvNet perception module, a manager, and a worker that collaboratively learns the hierarchical reward. The manager learns explicit rewards from the environment while creating sub goals for worker. In order to adapt to CTF, the ConvNet perception module was replaced with an autoencoder encoding the full CTF state space into an intermediate representation, similar to how ConvNet encoded Atari pixels into intermediate representation.



(a) Goal reward 1b (b) Goal reward 2b

(c) Policy 2b (d) Policy 1b

Figure 2: My complicated figure

Feudal network assumed that the sub-rewards followed a von-Mises-Fisher angular distribution with the main reward. Intuitively, this meant the sub goals were likely in the same general direction as main goal, e.g. move towards door in order to enter next room, door being the sub-goal. In the case of CTF, sub goals are more likely on the opposite side of the grid, thus we inverted the angular distribution so that the sub goal is opposite of the main goal.

Our implementation of feudal network was based on an implementation found on Github. During training, both the original and our implementation could not successfully converge. In figure **??** the loss of both worker and manager remained high and the policy gradient failed to reduce value function loss. Due to limited time, efforts on feudal network were eventually abandoned.

REFERENCES

1. Chevalier-Boisvert, M. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.
2. Hadfield-Menell, D., Milli, S., Abbeel, P., Russell, S. J., and Dragan, A. Inverse reward design. In *Advances in Neural Information Processing Systems* (2017), 6768–6777.
3. McGovern, A., and Barto, A. G. Automatic discovery of subgoals in reinforcement learning using diverse density. In *ICML*, vol. 1 (2001), 361–368.
4. Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *AAAI*, vol. 8, Chicago, IL, USA (2008), 1433–1438.