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 [6] with hierarchical elements to safely learn reward functions from expert demonstrations.

PROBLEM FORMULATION

Hierarchical model. 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, $\mathcal{T}$ is the transition model that is the probability of reaching state $s\prime$ given that action a is taken in state s i.e., $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, 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 ρ_i

IRL notation. Let $\mathcal{D}$ be a set of expert demonstrations $\{\tau_i\} \backsim \pi^\star$, where τ is trajectories of state-action pairs $\tau = \{s_1, a_1 ..., s_T, a_T\}$ for a sequential task until time T. The reward function can be recovered from trajectories τ and is parameterized by ψ and denoted as $r_\psi(\tau)$.

Safe Exploration. Since the τ is sampled from $\pi^\star$, it is plausible to assume that the expert demonstrations likely will lead to high utility behaviour in the training $\mathcal{MDP}$ for which it was designed. However, it is hard to verify that the expert actually captured the intended *true* reward function for the actual $\mathcal{MDP}$ that the inverse agent finds itself in. For example, if the 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].

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 or safe behaviour that the expert intended since no uncertainty is taken into account in the original IRL formulation. We will therefore distinguish between the proxy reward function $\mathcal{R}_{proxy}$ and the intended true reward function $\mathcal{R}_{true}$ (only known to the expert). 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.

Feudal networks. After reviewing hierarchical IRL literature, we further investigated feudal networks [5] since it learns sub-goals rather than explicitly hard-coding them.

METHODS

The MaxEnt IRL learns reward parameters that will match the feature counts (from the optimal policy satisfying the reward function) with the feature counts from expert trajectories. Given some feature over the state-space $\boldsymbol{\phi}(s)$, the reward function was defined as a linear function parameterized by $\boldsymbol{\psi}$. One observation that can be made for hierarchical tasks is that features representing goals and sub-goals are more likely to occur than all other features [4]. 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.

Data: D=$\{\tau_1, ..., \tau_T\}$
Result: $[\rho_1, \rho_2, ..., \rho_k]$
Initialize $\boldsymbol{\psi}$
while $\Delta\boldsymbol{\psi} < \epsilon$ **do**
  1. Compute $\pi(a|s)$ for $r(s; \boldsymbol{\psi})$;
  2. Solve for state visitation frequency $p(s|\boldsymbol{\psi}, \mathcal{T})$;
  3. Compute $\nabla_{\boldsymbol{\psi}} L(r(\circ; \boldsymbol{\psi}))$ from eq. 1;
  $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi}$ + $\nabla_{\boldsymbol{\psi}} L(r(\circ; \boldsymbol{\psi}))$
end
$\rho_i = \arg\max$ *r*$(\circ; \boldsymbol{\psi})$ *;*
if $\rho_i == s_0$ **then**
  return HIRL(split(D,ρ_i)),ρ_i, HIRL($split(D,\rho_i)$)
else
end
HIRL

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 maximum-likelihood function, and the resulting gradient can be shown to be:

$$\nabla_{\psi} L(r(\circ;\psi)) = \frac{1}{M} \sum_{\tau_d \in D} \sum_{s_d^t \in \tau_d} r(s_d^t;\psi)(\phi(s_d^t) - \sum_{s' \in S} r(s';\psi)\phi(s')$$

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

Safe Exploration. Further, 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 $\{\mathcal{P}^a_{ss'}\} = 0$ for each value iteration phase in the MaxEnt algorithm. The inverse agent should learn to distinguish the cases of *safety* (trust and treat expert trajectories as fixed) and *uncertainty* (distrust and divert from policy in case of uncertainty).

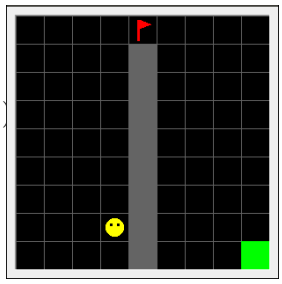
EXPERIMENTAL RESULTS

We evaluated our approaches in a object gridworld, using a modified interface for OpenAI Gym [1]. The agents move in the four cardinal directions ⟨up, down, left, right⟩. The terrain types are ⟨goal, flag, wall, road, lava⟩. The goal for the inverse agent is to first capture the flag (CTF) and then return to the base, while only traversing road cells and avoiding lava.

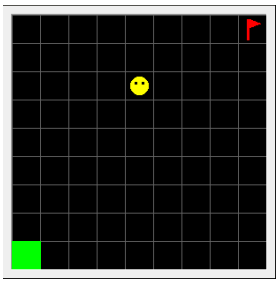
Inferring Hierarchical rewards. The results.

Safe Exploration. The results show that combining a safety criterion with risk-averse planning allow the inverse agent to avoid unforeseen scenarios. As seen in figure below, the agent diverted from the policy in the case of uncertainty, although the proxy reward function from the expert was not aware of lava and therefore did not include that such states should be avoided. By implementing the safety criterion, the true reward was fulfilled. 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.

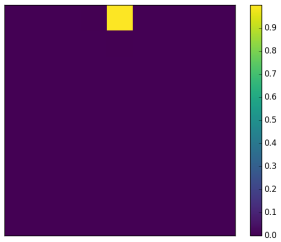
Feudal networks. The original feudal network consisted of a ConvNet for pixel perception, a manager, and a worker that collaboratively produces policy gradient. Feudal network assumes that the sub-rewards followed a von-Mises-Fisher angular distribution with the main reward. Intuitively, this meant the sub-goals were likely in the same general direction as main goal. In some CTF configurations flags are usually on the opposite side of the grid as the base, thus we inverted the angular distribution so that the sub-goal is opposite of the main goal. We based our implementation on an existing implementation [3]. During training, both the original and our implementation could not successfully converge and due to limited time it was eventually abandoned.



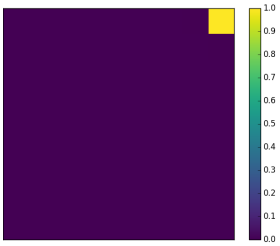
(a) Capture the flag A



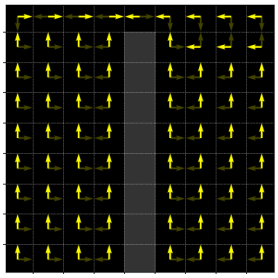
(b) Capture the flag B



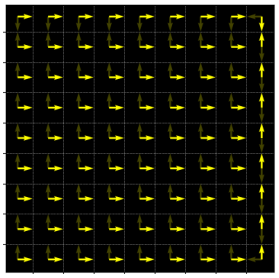
(c) Sub-goal $r_{\psi}(\tau)$



(d) Sub-goal $r_{\psi}(\tau)$

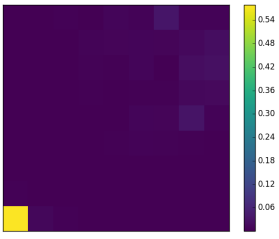


(e) Sub-goal π

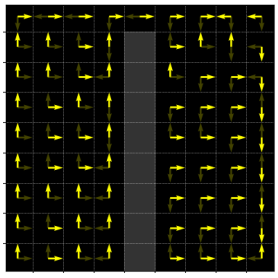


(f) Sub-goal π

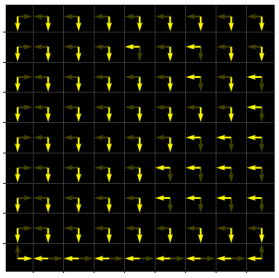
(a) Goal $r_{\psi}(\tau)$



(b) Goal $r_{\psi}(\tau)$



(c) Goal π

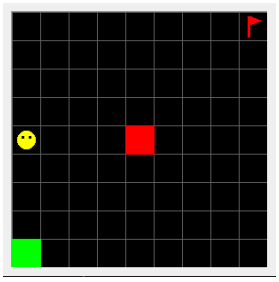


(d) Goal π

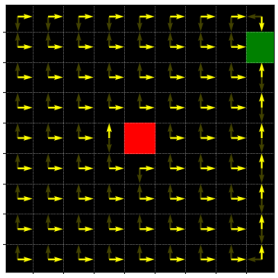
REFERENCES

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),



(a) *MDP* with lava



(b) Safe exploration achieved

6768–6777.

3. Makian, D. Implementation of feudal network. `https://github.com/dmakian/feudal_network`, 2017.
4. McGovern, A., and Barto, A. G. Automatic discovery of subgoals in reinforcement learning using diverse density. In *ICML*, vol. 1 (2001), 361–368.
5. Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. Feudal networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1703.01161* (2017).
6. 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.