Using Maximum Entropy Deep Inverse Reinforcement Learning to Learn Personalized Navigation Strategies

Abhisek Konar[1] and Bobak H. Baghi[1] and Gregory Dudek[1]

I. INTRODUCTION

Our work focuses on using inverse reinforcement learning (IRL) to produce navigation strategies where the policies and associated rewards are learned by observing humans. In particular, we are interested in developing intelligent agents that can mingle with people in crowded environments. While classic obstacle avoidance and navigation algorithms can be adapted to satisfy the basic navigational needs of such robots [1], [2], encoding the norms of social interaction has proven elusive. IRL provides a natural mechanism for learning social heuristics [3]. We present a deep IRL algorithm that simultaneously yields both the expert's underlying reward structure and its associated optimized policy.

II. PRELIMINARIES

Let the Markov decision process (MDP) $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma\}$ consist of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, transition dynamics $T(s,a) = P(s'|s,a)$, rewards $\mathcal{R} : S \mapsto \mathbb{R}$, and discount factor $\gamma \in [0,1)$. The IRL problem as formulated by Ng and Russel [4] is to recover an otherwise unknown and ambiguous $\mathcal{R}$ from trajectories of the form $\zeta = \{(s_0, a_0), (s_1, a_1) \dots\}$ sampled from a near optimal expert policy π^*. Ziebart et al. [5] resolve the inherent ambiguity in finding $\mathcal{R}$ by considering a maximum entropy distribution for the trajectories

$$P(\zeta_i|\boldsymbol{\theta}) = \exp\{\sum_{s\in\zeta_i} r(\mathbf{f}(s), \boldsymbol{\theta})\} \tag{1}$$

Where the mapping $\mathbf{f} : \mathcal{S} \mapsto [0,1]^k$ is a state feature vector of k elements, $\boldsymbol{\theta}$ are the model parameters, and $r(\mathbf{f}(s), \boldsymbol{\theta})$ is the reward function. As shown in Ziebart et al. [5], the optimal parameters can be found by maximizing log likelihood as in (2).

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, L(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{\zeta\in\mathcal{D}} \log P(\zeta|\boldsymbol{\theta}, T) \tag{2}$$

Wulfmeier et al. [6] propose a deep learning algorithm where the reward function is a neural network whose weights are updated using (3). This requires the MDP to be solved within each IRL loop and the obtained policy π to be compared with the expert policy in order to compute the necessary weight update gradient.

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = (\mu_{\mathcal{D}} - \mathbb{E}_\pi[\mu]) \cdot \frac{\partial}{\partial \delta} r(\mathbf{f}(s), \boldsymbol{\theta}) \tag{3}$$

Here, $\mathbb{E}_\pi[\mu]$ is the state visitation frequency (SVF) under the policy π and $\mu_{\mathcal{D}}$ is the SVF of expert demonstrations. The detailed deep learning algorithm, the dynamic programming algorithm used to obtain $\mathbb{E}_\pi[\mu]$, and the calculation of $\mu_{\mathcal{D}}$ are found in Wulfmeier et al.[6] and not replicated here due to space constraints.

III. DEEP IRL SOCIAL NAVIGATION

We present a deep inverse reinforcement algorithm with a simple feature design to replicate navigation behavior within an synthetic environment given trajectories from an expert. In our preliminary work we do this in a grid world, but plan to scale up to more realistic environments in near future. Deep reinforcement learning (RL) provides a powerful and general mechanism that should be of sufficient generality.

A. Gridworld Environment

We model the environment as a 2D gridworld with action set $\mathcal{A}_{gw} = \{\text{up}, \text{right}, \text{left}, \text{down}, \text{stay}\}$. The states are the agent's integer coordinates that are bounded (actions leading out of bounds are treated as a "stay" action), a single goal state, as well as a variable number of obstacles states exist in the environment.

B. Feature Representation

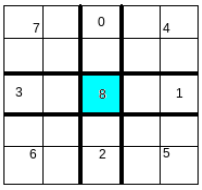
We craft a one-hot feature representation $\mathbf{f}_{gw}(s)$ based on both local and global information available to the agent, summarized in Fig. 1.

1) Goal Orientation: To encode orientation and enable reward learning based on the goal's location, The agent's 360° surrounding is split into b bins, which need not be equally sized. In our case, $b = 8$ and bins aligned with actions have a tighter angle (refer to Fig. 1).
We also consider three temporal features for moving away or towards the goal, or staying at the same distance from the goal which require previous state information to be calculable.

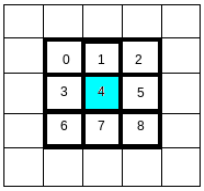
2) Obstacle Avoidance: To learn obstacle related rewards, we include the immediate $n \times m$ neighbourhood of the agent's current position in the features. This can be extended to allow rewards to be associated with respect to distance from obstacles. In our trials, we consider the 3×3 neighbourhood.

*This work was supported by Samsung Electronics.

[1]Abhisek Konar, Bobak H. Baghi, and Gregory Dudek are with the School of Computer Science, Centre for Intelligent Machines (CIM) at McGill University, Montreal, Canada. [akonar, bobhb, dudek]@cim.mcgill.ca



(a) *Goal orienting features used in our trials. each numbered area represents a bin.*



(b) *Local obstacle features. each numbered box is a potential obstacle location.*

Fig. 1: *Spacial IRL feature representations.*

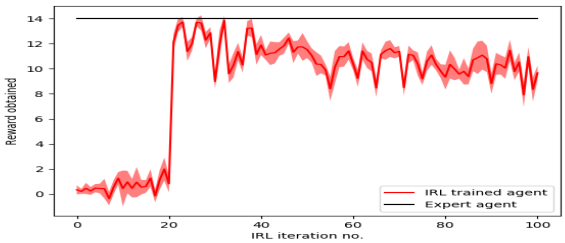
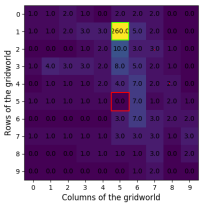
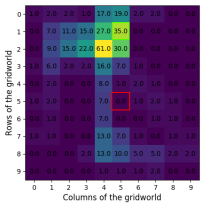


Fig. 2: *Average reward obtained by policy generated at each IRL iteration across 50 runs and 8 seeds compared with the reward obtained by the expert.*



(a) *Grid location visitation count of an unbiased agent.*



(b) *Grid location visitation count of an left-biased agent.*

Fig. 3: *For both plots, the agent performed 20 episodes of 20 steps each. The position of the goal and the obstacle are marked with a green and red square respectively. The starting position of the agent changed randomly with each run.*

C. IRL Algorithm

The Algorithm used is based on Wulfmeier et al. [6] with some key differences. Our method uses the actor-critic RL method [7] instead of the proposed approximate value iteration and does not assume known state transition dynamics. Instead, we use importance sampling to get the agent's SVF, where the probability of a trajectory being produced is assumed to be proportional to the reward it yields given the current reward function as shown in (4).

$$P(\zeta_k) \propto \frac{\sum_{s \in \zeta_k} r(\mathbf{f}(s)) - r_{min}}{Z} \tag{4}$$

Where r_{min} is the minimum reward obtained in the sample under the current reward function and Z is the partition function approximated by accumulating rewards for each sampled trajectory ζ_i as in (5).

$$Z \approx \sum_{\zeta_i} \sum_{s \in \zeta_i} r(\mathbf{f}(s)) \tag{5}$$

The visitation counts in each of the sampled trajectories are normalized to obtain the probability of a given state in a given trajectory. These normalized trajectories are then multiplied with their respective importance weights and summed to produce the state visitation frequency of the agent.

IV. Experimental Details

We evaluated the extent to which our IRL methodology could generate trajectories that would accrue as much reward as those our training examples, based on two classes of training data. One class was from an RL agent trained on the target environment using hand-crafted rewards (for obstacle avoidance and goal achievement). The second set was from human navigation, based on a desire to stay on the left or obstacles on the way to the goal. Our measure of success was the based on accruing as much reward as the training data, and also matching it's qualitative characteristics. The training sets sizes were 2400 and 30 samples respectively.

For both cases, we ran our IRL algorithm for 100 iterations (convergence was common around the 40^{th} iteration). The inner RL loop ran for 6000 episodes to guarantee convergence.

In both the cases the IRL agent was able to pick up the underlying reward the expert was trying to maximize and optimize for it. Fig. 2 shows that the performance of the IRL agent after roughly 25 iterations is comparable to that of the expert and Fig. 3b show a noticeable shift in trajectories of the biased agent compared to the unbiased one in Fig. 3a.

V. Conclusions

In this work we showcased preliminary results from our IRL-based method for learning human-like navigation behavior. The simple features with a deep reinforcement learning architecture is promising and further research will investigate this method's scaling to larger environments and learning from human trajectories.

References

[1] J. H. Reif and H. Wang, "Social potential fields: A distributed behavioral control for autonomous robots," *Robotics and Autonomous Systems*, vol. 27, no. 3, pp. 171–194, 1999.

[2] J. Rios-Martinez, A. Spalanzani, and C. Laugier, "Understanding human interaction for probabilistic autonomous navigation using risk-rrt approach," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 2014–2019.

[3] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 35, no. 11, pp. 1289–1307, 2016.

[4] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, vol. 1, 2000, p. 2.

[5] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." 2008.

[6] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum entropy deep inverse reinforcement learning," *arXiv preprint arXiv:1507.04888*, 2015.

[7] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.