Breaking the Tie: Evaluating human preferences in Reinforcement Learning

by

Tuhina Tripathi

B.Tech., Delhi Technological University, 2019

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Computer Science

2023

Committee Members:

Dr. Bradley Hayes, Chair

Dr. Alessandro Roncone

Dr. Nikolaus Correll

Tripathi, Tuhina (M.S., Computer Science)

Breaking the Tie: Evaluating human preferences in Reinforcement Learning

Thesis directed by Assistant Professor Dr. Bradley Hayes

As robots begin to work alongside humans, it becomes crucial for them to understand their task and objectives. Specifying these objectives in the form of rewards is challenging and might not be able to model the user's preferences. Prior work on preference learning defines preference in a comparative manner, in the form of either absolute or relative choices. However, human preferences can be sub-optimal and this can cause the agent to end up in risky situations. Therefore, it is important to recognise when preference is acceptable through a methodical evaluation in order to learn preferences safely. This work builds upon a definition of preference that identifies it as a tie-breaker: an underlying factor that makes the human arrive at a diverse reward function than the robot agent [39]. The main contribution of this work is to operationalize an evaluation approach to determine acceptability of human preference and obtain notions of bounded risk in learning from preferences. Inverse reinforcement learning is used to learn the user reward function from of demonstrations that encode human preference. The obtained user policy is compared with the optimal policy. We define a threshold for risk based on the distribution of expected rewards returned for the optimal policy and evaluate different user policies for acceptability. We implement the proposed approach in two task environments - a grid game and OpenAI gym 'highway-env'. We calculate the threshold for risk for each of the given tasks. Our results show that the approach is able to identify which policies that are acceptable given our tolerance for risk. It is also able to remove failed demonstrations based on the evaluation. We show that the final policy learned after removing demonstrations that are out of the risk bounds, is able to avoid risky states in the environment.

Acknowledgements

First and foremost, I would like to thank my advisor Dr. Bradley Hayes for the constant guidance and support over the past two years. I started with very minimal knowledge of robotics but Brad helped me maintain a positive outlook and see the bigger picture throughout my work. It was a pleasure learning from and working with him.

I would like to thank members of the Collaborative AI and Robotics Laboratory for their constructive feedback, great discussions and especially their friendship. This work would not have been possible without your support.

Finally, I would like to thank my family and friends for believing in me and supporting me throughout my research.

Contents

Chapter

Figures

Figure

Chapter 1

Introduction

Over the past few years, there has been an enormous effort towards incorporating robots and AI-systems into our daily lives. While these agents have been very effective in improving the overall efficiency and productivity of the system, their successful integration in the real-world requires them to interact with, learn from and adapt to humans and other AI-agents in an intelligent manner [11]. In real world collaborative tasks, it is crucial for these agents to have the ability to acknowledge the influence of human preferences on their behavior and align their perception of the environment with that of the human.

To achieve human-like interaction, robots should be able to comprehend the task objective. Specifying these objectives manually, in the form of rewards, is extremely challenging and also has the risk of reaching catastrophic outcomes [17] [3]. A better approach to learn these objectives is from humans themselves. There has been a lot of outstanding work on learning from demonstrations for a task [8] [4] . In recent years, researchers have also focused on learning human preferences from demonstrations. However, most of the existing methods work with a comparative definition of preference. Preference, in these methods, is defined as a relative favourability for one or more options. Some approaches use actively querying the human while training a model to get pairwise trajectory comparisons [30] [12] and maximize the likelihood of generating the preference-encoded behaviour [35] [19] . Some other approaches make use of relative and absolute comparisons like ranking of trajectories and asking the demonstrator for numerical ratings [10] [9]. Generating informative queries for comparative feedback is a problem to be tackled when using the active-

learning approach. In the case of absolute preferences, trying to assign an underlying numerical value to a demonstration is non-trivial and sometimes unintuitive. Also, these approaches require the demonstrator to have a certain level of proficiency in the task, which might not always be possible. Recent work also proposes a regret preference model [18] which models a segment's desirability based on the negative difference between the segment and the optimal policy's return in each transition. Our work proposes a methodical definition of preference as a tie-breaker and builds upon it to learn preferences safely.

There is very limited literature on evaluating user preferences and learning notions of risk for preference learning. There can be scenarios in which human preferences are unreasonable resulting in sub optimal demonstrations being given by the human demonstrator. Not evaluating human preference can lead to undesirable outcomes, introducing a risk factor in learning from preferences. Recent work talks about how inferring rewards from demonstrations and preferences can cause relearning failures [22]. Some work tackles sub optimal demonstrations by removing trajectories for which the expected returned reward is below a certain performance threshold [32] [15]. Some other work[6] estimates the expertise of the demonstrator and filters out sub-optimal behaviour. By removing unsuccessful trajectories, these methods are able to mitigate the risk. However, failed demonstrations also carry important information and can aid in improving the learning process. Some prior work assumes that the human and robot have a mental-model disparity[34] and uses reward-coaching to mitigate the risk of incongruous models. But in our work, we assume that human has complete knowledge of the environment and any disparity between the rewards of the human and agent is solely due to human preference.

In this work, we propose a new method for evaluation of preferences. Our method is based on the definition of preference as a tie-breaker. Originally introduced in [39], the work defines preference as an underlying factor that results in the human and agent having different reward functions. In section 3.2 we describe an an intuitive real-world example of how preference acts as a tie-breaker in scenarios when the human perceives two options as the same, which in reality have different costs. Our approach of learning preferences does not use explicit comparisons. Instead, we

assume that the demonstrated behaviour is actually the preferred behaviour for a human. Inverse reinforcement learning is used to learn the reward function for the human and then the recovered policy from this reward is evaluated. We base the evaluation of human policy on the distribution of expected return of rewards of the optimal policy (Section 3.6). We also present experiments, to prove the merit of the approach, on two environments - a grid game and a driving simulation.

Towards operationalization of an evaluation model of preference, we first discuss the related work in the field in Chapter 2. We then describe the proposed approach, in terms of the problem formulation, preference definition used, and the methods used to evaluate the preference. In Chapter 3, we dive into our experiments and also present an analysis for different example user policies. Chapter 5 and 6 present our results and conclusion respectively. Lastly, we talk about some interesting future directions to explore for this work.

Chapter 2

Background and Related work

2.1 Reinforcement Learning

Reinforcement Learning or RL is a machine learning technique that aims to learn optimal behavior in a dynamical system (Sutton and Barto, 1998)[33]. A simple model of RL is shown in Figure 2.1. It uses a Markov decision process (MDP) to model the environment. An MDP is a tuple $M = (S, A, T, R, \gamma)$ where $\mathcal{S}$ is the state space that represents different situations in the environment the agent might end up in. $\mathcal{A}$ denotes the set of possible actions the agent can take in the environment. We have a transition probability function $\mathcal{T}$ which defines how the system evolves. If the agent is at state s_t and takes an action at at the given time step 't', it can transition to a new state s_{t+1} based on :

$$S_{t+1} = t(.|s, a)$$

The reward function $\mathcal{R}$ defines the rewards an agent can receive at every step. For instance, the agent might receive a higher reward in states closer to the goal states and lower rewards when it's further away. γ is the discount factor which defines the importance of future rewards. A policy π is a mapping of the states to the actions which describes how the agent will behave in the environment. In RL, the goal is to learn behavior that can maximize the expected cumulative reward an agent can receive in the environment. The policy which maximises the return of rewards is called the optimal policy and is denoted by π^*. It maps every state to the best action that can be taken.

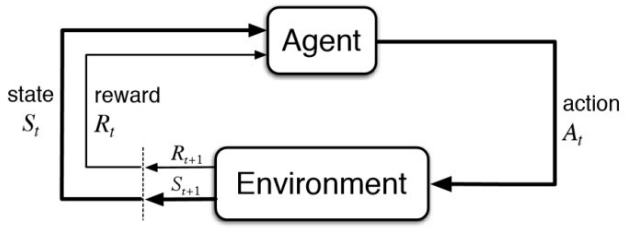

Figure 2.1: Agent-environment interaction in reinforcement learning

The most widely used methods for Reinforcement learning are value-based methods such as Q-learning and Deep Q-networks (DQNs) [23], policy-based methods such as deep Deterministic Policy Gradients (DDPG)[20], Actor-Critic [36] and Proximal Policy Optimization(PPO)[31], amongst many others.

2.2 Inverse Reinforcement Learning

Inverse Reinforcement Learning(IRL) is a technique that aims to learn the objectives of a task using demonstrations provided by an expert. The input to an IRL algorithm is a set of trajectory demonstrations $D = \{\tau_1, \tau_2, ..., \tau_n\}$ given by an expert which are optimal for the given task. The goal is to learn the underlying unknown reward function [1] [24] [25] [2].

IRL is a powerful approach for solving many real-world problems. If we can infer the underlying reward for a task, we can perform standard reinforcement learning with the learned reward function to control the system even in the absence of an expert. It is also very useful in many real-world scenarios, for example, developing an autonomous vehicle which needs to predict the actions of other vehicles on the road, also known as Behavior Modeling [21]. In order to effectively interact with other vehicles, the agent should be able to understand the objectives of the other cars.

Some of the most influential work in IRL has been Maximum Margin Planning(MMP) [29], Apprenticeship Learning[1] and Bayesian Inverse Reinforcement Learning[28]. In many IRL scenarios, multiple possible reward functions can explain the observed behavior. Ziebart et. al. proposed

a solution for this ambiguity problem by introducing Maximum Entropy Inverse Reinforcement Learning [40]. This approach assigns probabilities to reward functions based on their likelihood of generating observed behavior and is a very popular method in IRL. Many real-world domains have high-dimensional state spaces and hence the need to learn complex representations of data. With this in mind, Wulfmeier et al. proposed an approach to use deep learning techniques to approximate the reward function, Maximum Entropy Deep Inverse Reinforcement Learning [38].

In standard IRL approaches, the goal is to learn rewards from expert demonstrations. However, in many problems, specifically in robotics, expert demonstrations may not be available. There are several reasons for this 1) good demonstrations require a certain level of expertise 2) it is difficult to manually operate robots (especially with high degrees of freedom). These problems highlight that even though demonstrations can provide a lot of useful information, it is not always sufficient.

Some alternatives to this are learning from sub-optimal demonstrations which involves learning from failed demonstrations [15] [32] or incorporating other sources of information such as corrections [5] or preferences [13] [16] [37] [30] [26]. This work is related to IRL as, similar to IRL, human demonstrations encode preference information and learning a reward function from these demonstrations ensures that the agent will perform in accordance with the human's preferences.

2.3 Preference Learning

Preference-based learning refers to techniques that involve incorporating human feedback in order to enable an agent to make decisions that align with human expectations. This is an active area of research. Some methods make use of relative comparisons for learning preferences. Active preference learning involves the learning agent actively generating pairwise preference queries for a human demonstrator to label [13] [30]. Trajectory-ranked Reward Extrapolation(T-REX) uses a set of pre-ranked demonstrations to learn human preferences [9]. Other methods approach this problem using absolute preferences involving quantitative ratings of demonstrations [10] [14].

Learning human preferences is a challenging problem due to many reasons: 1) Human preferences are often ambiguous and multifaceted, making it difficult to design relevant queries. 2)

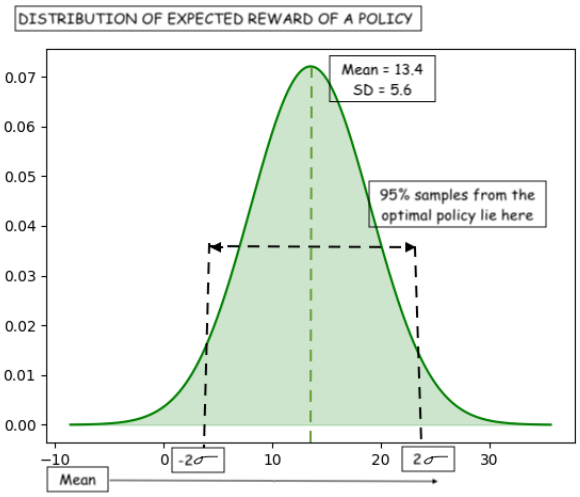

Figure 2.2: Distribution of expected return of rewards for a policy

Getting good quality data requires the presence of an expert which is not always feasible 3) Preference labels for comparative feedback is very expensive and time-consuming.

In this work, preferences are not learned explicitly in the form of feedback. Instead, we assume that demonstrations encode preference information i.e. the demonstrated behaviour is the human preferred behaviour. This enable us to reason about a user's decision making process directly through the demonstrations given by the user.

2.4 Distribution of expected return of rewards of a policy

The expected return of reward of a policy is the mean reward an agent can estimate to get from multiple roll outs of the policy in the environment. A distribution over the expected reward can help analyse the performance of a policy, as it gives an idea of the range of rewards that can be returned and also the mean around which most returned rewards are concentrated. It can be used to compare different policies in terms of their mean and variance, as is the case in our work. We compare policy distributions of the user policy with the optimal policy in this work, to get a sense of how close the distributions are.

An intuitive example to explain this concept of comparing distributions is as follows. Suppose

we have an optimal policy in a stochastic environment. The optimal policy gives the best action for each state but due to the uncertainty in the environment, the agent ends up in different states with some probability. When multiple roll outs of the policy are run, the returned reward will varies with each run, converging towards a mean value as the iterations are increased. Plotting a distribution of the returned reward gives a notion of performance of the policy. In this work, we use the distribution of expected reward of the optimal policy to calculate bounds of risk for a user's performance. The maximum risk we can incorporate in learning human's preference is the worst case performance of the best policy. This is shown in figure 2.2

Chapter 3

Technical Approach

3.1 Problem Formulation

We formulate the environment as a Markov Decision Process with a set of states $\mathcal{S}$, possible actions $\mathcal{A}$, transition probabilities $T : S \times A \times S \to [0, 1]$, a reward function $R : S \times A \to \mathbb{R}$ and a discount factor denoted by γ. Policy π is a mapping of the states to the actions which describes how the agent will behave in the environment. For an MDP under a given policy, the expected return is given by:

$$J(\pi|R^*) = E[\sum \gamma^t R^*(s_t, a_t)]$$

For our problem, we also assume that we have an MDP without a reward function $MDP\backslash\mathcal{R}$. We instead have access to a set of demonstrations $D = \tau_1, \tau_n, ...\tau_n$ which consists of trajectories of state and action pairs ($\tau_i = s_0, a_0, s_1, a_1, ...$). The demonstrator is assumed to be optimising an internal reward function $\hat{R}$. The goal is to learn preference from human demonstrations while also making sure that the preference is compatible with the task environment.

3.2 Preference as a tie-breaker

In this work, we define preference as a tie-breaker, an underlying factor that leads to difference in reward functions of the human and agent. Preference is considered to be the cause that leads the human to have a difference in understanding of the environment from the agent.

We introduce the following example to better explain the concept of preference as a tie-

breaker. Suppose you are getting to your workplace from home and you wish to grab a coffee on the way. You have two options for the coffee shop you can go to, A and B, which are both on two different equidistant routes. Both shops have the same selection of products and prices and initially, are at the same distance along the route. There is no quantitative difference between the two options. But shop A plays really loud music in the morning, which is the only difference. You wish to have a calm morning so you prefer going to shop B instead of A. Now both options were equivalent but preference was the tie-breaker in this situation, that lead you to choose B (preference for a calm morning). What happens if B is a longer detour as compared to A? Assuming there is a difference of δ between the two paths, i.e., the cost of taking path with coffee shop B is δ more than the coffee shop A. If you still choose to go to shop B, we say that you perceive both options as equivalent but still choose to prefer B. But an interesting case arises if B is a really long detour, as compared to A, the difference in costs δ is a much larger value. Would choosing B still be reasonable?

The idea is to understand at what value of δ does the human preference become unreasonable. We want to find a threshold value below which, preference cannot be seen as a tie-breaker and the human preference can actually be classified as incompatible with the task environment.

3.3 Terminology

In order to explain the approach better, we define the following terms:

- $\mathcal{R}$: denotes the ground truth reward function
- $\hat{\mathcal{R}}$: denotes the reward learned through IRL from human demonstrations. $\hat{\mathcal{R}}$ models human preference
- Tie: A tie between rewards occurs either when the expected rewards are actually equal OR perceived as equal by the human
- δ: A threshold (distance from expected reward of an optimal policy) within which human preference is termed to be compatible with task environment

We now list the different scenarios under which preference can be defined and assumptions we consider in our work:

- Under situations in which $\mathcal{R} = \hat{\mathcal{R}}$ we say that the human and agent interpret the environment as exactly the same, and preference does not exist (because the human does not value one path over another, otherwise it would surface in their reward function)
- Preference can only exist if $\mathcal{R} \neq \hat{\mathcal{R}}$
- The human is assumed to have complete knowledge of the environment

3.4 Learning optimal policy for the environment

To learn the optimal behaviour of an agent in a dynamical environment, we make use of standard reinforcement learning techniques. There are many different methods for Reinforcement Learning but for this work we are mostly interested in Q-learning and Deep Q-networks.

3.4.1 Q-learning

Q-learning is a value-based, model-free algorithm that aims to find the best actions an agent can take in every state so as to maximise the expected sum of rewards. The main idea is to build a Q-table of state-action values having the dimensions $\mathcal{S} \times \mathcal{A}$ and iteratively update by acting in the environment. The algorithm finds a mapping of each state and action pair in the environment to a Q-value.

Q-learning uses the Bellman equation to update values for each state-action pair:

$$Q(s, a) = E[r + \gamma max(Q(s', a'))|s, a]$$

Here $Q(s, a)$ denotes the value of taking action 'a' in state 's', E denotes expectation, r is the actual reward the agent receives in that state, γ is the discount factor that decides the importance of future actions. The term $maxQ(s', a')$ is the value of the best action that can be taken in the

immediate next step s'. Therefore, γ multiplied by the value of the next action allows for weighting of future rewards; if gamma is closer to 1, we give more importance to future rewards and vice-versa. The algorithm for q-learning is shown in Algorithm 1.

In many real-world scenarios, the state space is huge making Q-tables intractable to maintain and update. Deep Q-networks can be used in this case to approximate the table using neural networks.

3.4.2 Deep Q-networks

Deep Q-networks model the state-action values as a Q-function rather than a Q-table. DQNs make use of neural networks to model the Q-function. The underlying idea is the same as that in Q-learning. The algorithm iteratively updates the estimate of Q-values by acting in the environment.

The architecture of a DQN is shown in Figure3.1. It has two neural networks - the Q-network, target Q-network, and a database buffer called Experience Replay. The agent interacts with the environment and collects training data that is stored in the Experience Replay. From this training data, a batch of recent and older samples is taken and sent to the two networks for training. The Q-network predicts q-values for current states and actions from the batch whereas the target Q-network takes the next states and actions and predicts the q-values for those. These q-values and the observed reward at the current time step are used to train the Q-network. Every few time steps, the weights of the Q-network are copied to the target network.

3.5 Learning reward from human demonstrations

In this work, we use Inverse Reinforcement Learning to learn a user's reward function. We define an MDP without a reward function $MDP \backslash R$. We assume access to a set of demonstrations $D = \{\tau_1, \tau_2, ...\tau_n\}$. Standard IRL algorithms assume access to an expert's demonstrations to learn the optimal reward function for the task environment. For this approach, we consider demonstrations that are sub-optimal as well.

The Maximum Entropy IRL approach is used to learn the human's reward function $\hat{R}$. The

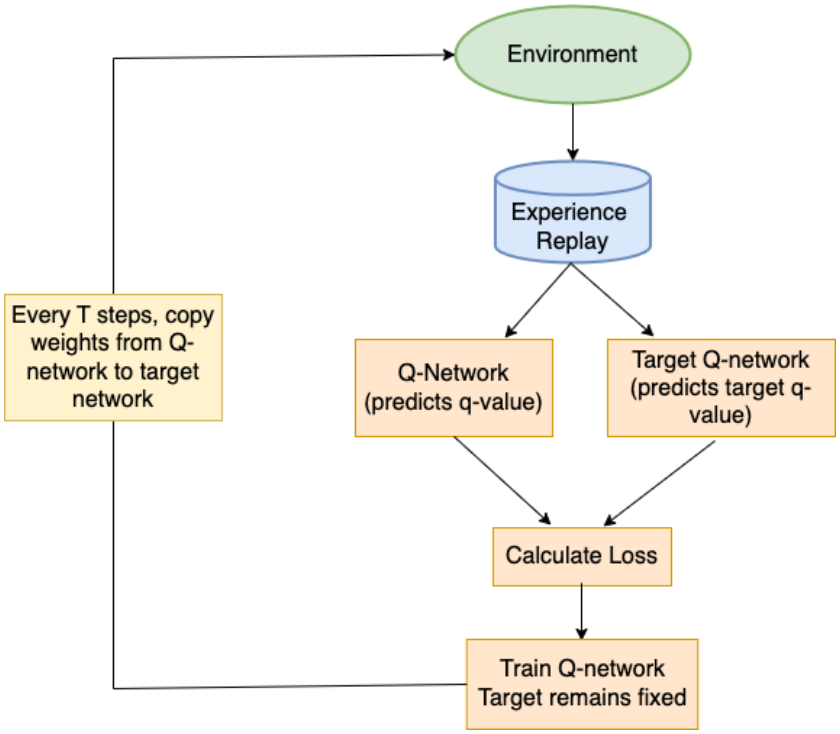

Figure 3.1: Architecture of deep Q-networks

advantage of using maximum entropy is the ability to solve the ambiguity problem i.e. handle different distributions over the trajectories. Using deep learning allows us to model complex non-linear reward functions for high-dimensional state spaces. The detailed algorithm is given in 1 [38].

To recover the reward function, we learn a feature representation $\phi(s)$ for all the states. We use hand-crafted features that give a sense of how good or bad a certain state is. We do not learn a modeling for the reward for each state. The structure that this paradigm follows is that states with similar features will have similar rewards. The reward is learned as a weighted combination of these features, where w^* denotes the optimal weights:

$$\mathcal{R} = w^*.\phi(s)$$

A neural network is used to map state features to the rewards. It consists of two fully-connected layers and uses a sigmoid activation function. The main idea is to minimise the loss between the visitation counts of the expert and the learning algorithm. For this, first the state action frequencies for the expert μ_D are calculated. We start with some random initialization of the weights θ^1 and estimate the reward for the given weights (parameters to optimise the neural network over). Then the MDP is solved for the given rewards to obtain the policy π^n at the current iteration. This policy is propagated in the environment and the state-action frequencies are calculated for the current iteration. After that, we determine loss as the difference in the expectation of features for the expert and agent in the current iteration. Finally, we update the weights given the current weights and the loss. Once we get the optimal weights for the task, the reward is simply a dot product of the weights with the features.

3.6 Evaluating human preference

The algorithm in 1 helps us recover a reward from a set of user demonstrations. We use standard RL techniques to get the policy from the reward function. Since we do not assume that the

Algorithm 1 Maximum Entropy Deep Inverse Reinforcement Learning

Input: $\mu_D, f, \mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma$
Output: Optimal weights θ^*
Initialize weights $\theta \rightarrow \theta^1$
Iterate: $n = 1 \rightarrow$ max_iterations
$r^n = \text{nn\_forward}(f, \theta^n)$

$\pi^n = \text{solve\_mdp}(r^n)$
Calculate $E[\mu^n]\ from\ \pi^n$

${L_D}^n = log(\pi^n) \times {\mu^a}_D$

$\frac{\partial {L_D}^n}{\partial r^n} = \mu_D - E[\mu^n]$

$grad = \text{nn\_backprop}(f, \theta^n, \frac{\partial {L_D}^n}{\partial r^n})$

$\theta^{n+1} = \text{update\_weights}(\theta^n, grad)$

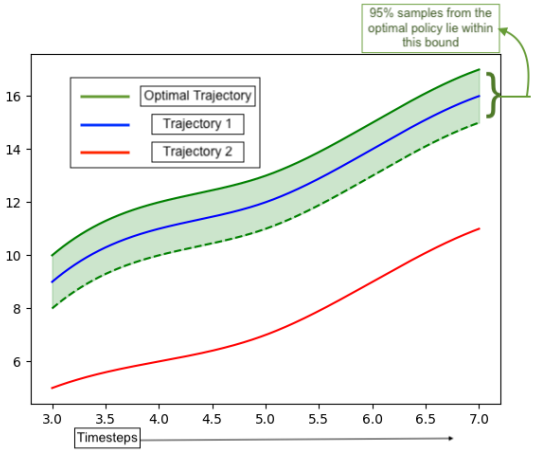

Figure 3.2: User policy evaluation based on the distribution of the expected rewards of the optimal policy.

demonstrations are sampled from an optimal policy, so that they can encode preference information, the policy learned from these demonstrations might encode some sub-optimal behaviour. We need to evaluate the preference-encoded user policy to ensure that it does not exhibit unreasonable or risky behaviour.

We base our evaluation of the user policy on the distribution of expected rewards of the optimal policy. We define δ as a threshold for risk which refers to the worst case performance of the best policy. Given an optimal in a stochastic environment, we generate a large number of roll outs from the optimal policy and plot the distribution of the returned rewards. The upper bound for delta is naturally the return of the most optimal rollout. The lower bound for delta is the value within which 95% of the samples from the optimal policy lie. Figure 3.2 is a representation of this idea. An important thing to note here is that we're are comparing a single trajectory to a distribution which is only for illustration, the actual evaluation compares distributions of the user and optimal policy. Trajectory-1 sampled from a given user's policy would be considered acceptable because it lies within the risk bound δ for the task. Whereas, trajectory-2 sampled from the same or perhaps different user policy is unacceptable for this evaluation as it lies outside of our risk bounds.

Chapter 4

Experiments

We access the performance of our proposed approach in two environments, a grid game and the OpenAI gym highway-env. We model the environments as an MDP with a true reward function $\mathcal{R}$. The environments have a stochastic transition probability function. Reinforcement Learning is used to first find an optimal policy π^* in the environment and then we use the MaxEnt Deep IRL described in section (number) to recover a reward function $\hat{R}$ given the user's demonstrations $\mathcal{D}$.

4.1 Grid Game

The grid game MDP is a 5×5 grid. The agent's state is a specific cell in the grid. The possible actions for the agent are moving one step in one of the four directions $\{Right, Up, Left, Down\}$. The episode can terminate at either the final reward state of +20 or the failure state of -3. The reward +1 is not a terminal state and is in between wall states to force the agent to take the in between in order to get the maximum reward. The first row of the grid is lava, having non-terminal states with reward of -1. Each step has a penalty in order to incentivise shorter paths. The task for the agent is to get the maximum reward in the game in the least number of steps. Figure 4.1 shows the task environment.

4.1.1 Learning the optimal policy

We first learn an optimal policy for the task using Q-learning. The q-learning algorithm uses an epsilon-greedy policy to find the q-values for each state and action. We use an exploration rate

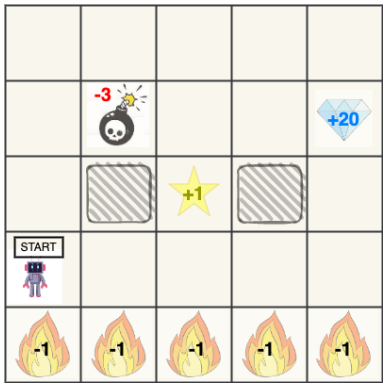

Figure 4.1: The grid game task environment

of 25% ($\epsilon = 0.25$). The discount factor $\gamma = 0.9$ and the learning rate $\lambda = 0.1$. Q-learning gives a mapping of states to the best actions for the task.

4.1.2 Recovering reward from demonstrations

Next, we generate demonstrations in the environment with different user preferences. The demonstrations are a set of trajectories consisting of the grid cell (0 to 24) and corresponding action (one of $\{Right, Up, Left, Down\}$). In order to recover the user's reward $\hat{\mathcal{R}}$, we define a feature vector for user demonstrations. The feature vector is a one-hot encoded vector for each grid cell. We calculate the feature expectations for the demonstrations as the state visitation frequencies for each state in the task.

We consider different example scenarios to evaluate human preference. The first case we consider is of a user that is only able to get the +20 reward and does not get the +1. The trajectory does not get the highest reward for the task but still completes the task successfully. The case is of a failed demonstration in which the user crashes into the bomb (-3 reward terminal state) and is unable to complete the task. Apart from these extremes, we also consider a case in which the user follows a slightly longer trajectory but is able to get both the positive rewards and complete the task. The reward recovered for each of these cases is shown in figure 4.2.

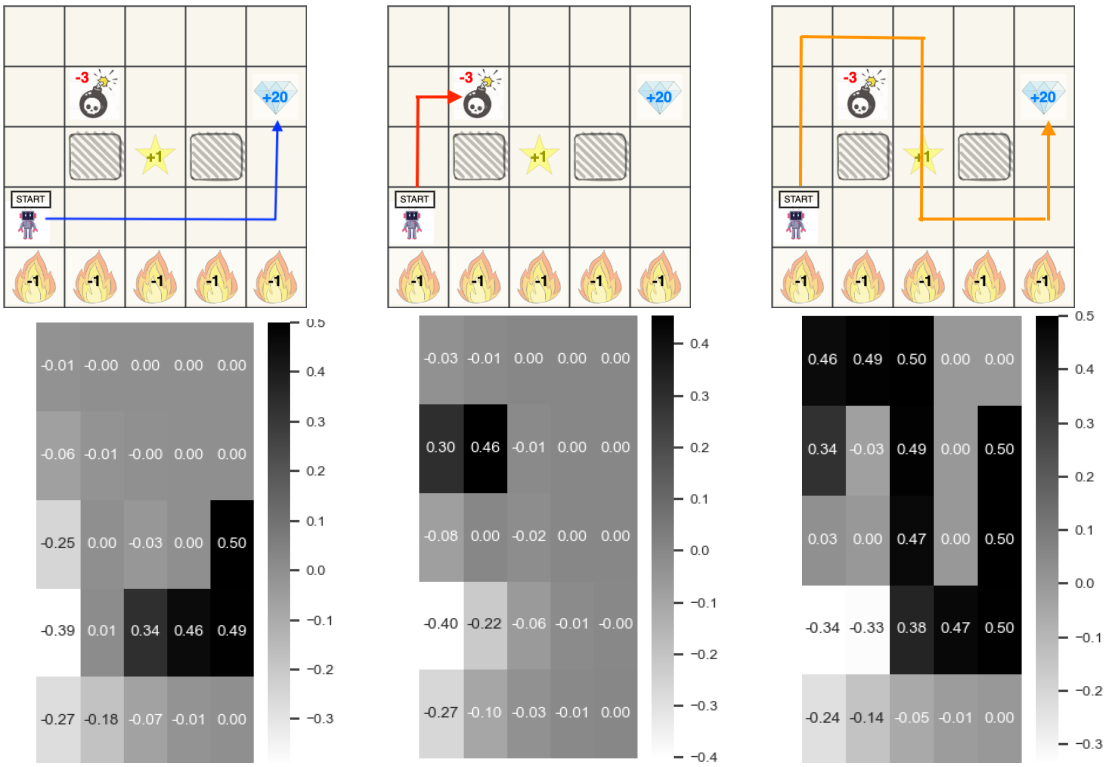

Figure 4.2: This image shows examples of trajectories sampled from three different user policies in the grid game along with the recovered reward (learned through IRL) for each.

4.1.3 Evaluation of User Policy

The recovered reward is used to learn the user's policy. Each policy is used to generate multiple roll outs in the stochastic environment. A distribution of the expected reward returned is plotted against the distribution of rewards for the optimal policy for evaluation. We present a case by case evaluation of the three user policies mentioned in section 4.1.2.

- **Case-1:** In this case, the distribution of returned rewards for the user policy shows that it lies within our risk bounds for the task. Therefore, we say that the given user policy is acceptable throughout.

- **Case-2:** For the second case, the user policy exhibits failure in completing the task. A plot of the distribution of expected returned reward for this policy along with the optimal policy returns shows that the policy lies outside of our risk bounds. This policy is unacceptable according the proposed evaluation metric. This shows an important property of the proposed algorithm that it is able to remove failed demonstrations as well.

- **Case-3:** The third case is of a non-optimal policy that is able to get both rewards and reach the goal position sometimes but takes a longer path to get to the goal. The comparison of the expected rewards returned somewhat overlap with the distribution for the optimal policy. In this case, we cannot directly state if the policy is acceptable or not. We learn from parts of the policy that lie within the risk bounds and remove parts that are outside of it. This ensures that we are incorporating human preference while also making sure that it doesn't not lead to unsafe choices.

Figure 5.4 shows comparison of distribution of the rewards returned for the user policy in the grid game compared against the distribution for the optimal policy.

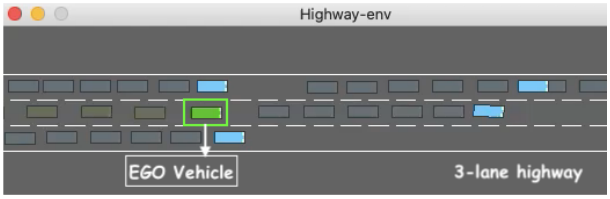

Figure 4.3: OpenAI gym highway-env

4.2 Highway Driving Simulation

We also implement the proposed algorithm in the OpenAI gym 'highway-env' which is a highway driving simulation environment. The particular environment has a three-lane highway with one user controlled vehicle (EGO vehicle) and multiple other vehicles whose behaviour is randomly generated. It has a continuous state space and stochastic transition function. The 'highway-env' models the non-linear stochastic highway driving in a deterministic way. So we introduce some stochasticity in the model by introducing randomly sampled noise in the dynamics of the vehicles. We sample random noise in the speed and angular acceleration of the vehicles in the environment.

The environment is modeled as an MDP with the following properties:

- **States:** The state space is continuous with each state being a $V \times F$ array describing of a list of V nearby vehicles by a set of features $F = \{$"x", "y", "vx", "vy", "$heading$"$\}$.
- **Actions:** The action space is discrete with the following actions: $\mathcal{A} =\{$ 0: 'LANE_LEFT', 1: 'IDLE', 2: 'LANE_RIGHT', 3: 'FASTER', 4: 'SLOWER'$\}$. These actions type adds a layer of steering and speed control over the continuous low-level control.
- **Transitions:** The transitions follow the bicycle model and are stochastic because of the noise introduced in the dynamics.
- **Rewards:** The agent gets a high reward for the following behaviour: 1) appropriate speed and distance from others 2) less lane change and 3) no collisions. A reward wrapper is used to incorporate mentioned reward features.

The goal for the EGO vehicle is to run in the simulation for 30 time steps without collision. There is no idea of a goal state in this case but the agent gets a positive reward of +10 on completing 30 seconds without collision.

4.2.1 Learning the optimal policy

The Stable Baselines3[27] deep Q-network implementation is trained on the 'highway-env' to get the optimal policy for the environment. The library provides a vanilla DQN implementation [23] using convolutional neural networks to approximate the Q-values for each state. The DQN uses the 'MlpPolicy' to extract features from high-dimensional observations. This 'policy' is different from an RL policy, it is simply a class of networks used to predict actions.

4.2.2 Recovering reward from demonstrations

Demonstration trajectories are generated from different user policies using keyboard controls by manually driving the vehicle in the simulation. Each trajectory is a sequence of $(state, action)$ pairs for 30 time steps. To learn the reward function from the demonstrations, we define a vector of hand-crafted features. The feature vector is (9×1) matrix with the following features describing the state of the vehicle:

- distance from neighbouring vehicles in each lane (6 features)
- speed : rewarded between a range
- heading : ensure vehicle moves in a straight line
- collision : (0/1) for collision

The reward is a linear combination of the above-mentioned features: $\mathcal{R} : \theta^* \cdot \phi(s)$
The goal is to find the optimal weights θ^* that satisfies user's intended behaviour. We use Maximum Entropy IRL to learn the optimal weights for the features.

4.2.3 Evaluation of User Policy

Having learned the optimal weights for the user's policy, we can use that to recover reward function. We use the same procedure as we did for the grid world and determine the user policy from the recovered reward, generate roll outs and plot the distribution of rewards returned. We evaluate the following two user polcies:

- **Case-1:** We first evaluate the case of a "bad" driver. Some peculiar properties of these trajectories are speeding, not maintaining safe distance from other vehicles, frequent overtaking of other vehicles and eventually resulting in a crash. The distribution of rewards for the user's policy lies outside our bounds of risk and therefore, we claim that user policy is unacceptable. This is a case of failed demonstrations which our approach is tackle.

- **Case-2:** We consider the case of another driver who does not exhibit optimal behaviour but does not cause collisions either. The trajectories involve occasional speeding, lane changing and sometimes not maintaining safe distance from the other vehicles. A comparative analysis of this user's policy shows some overlap with the optimal, which shows that the user policy is acceptable in a given region (within the bounds of our risk).

We discuss the results of these different cases in Chapter 5.

Chapter 5

Results

We analyse the different cases of user preference in the grid-world and highway driving environment. As seen in Figure 4.2, our evaluation is able to differentiate when user preference is acceptable or unacceptable. Another promising result is that it is able to identify and remove failed demonstrations from the learned policy. We further analyse the Case-3 user trajectory in the grid game to identify how the evaluation is able to stay within our bounds of risk.

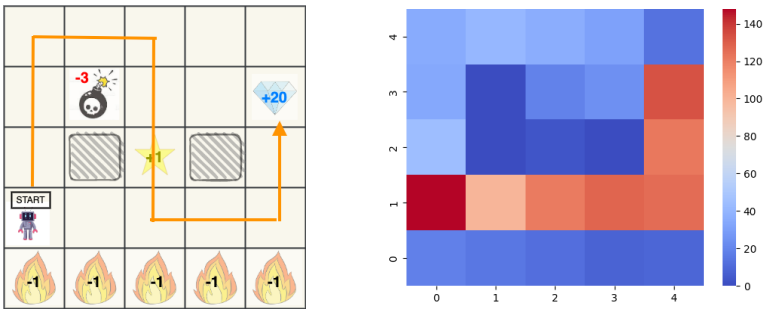

Figure 5.1: A heat map showing state visitation frequencies for the user policy after evaluation. The left image shows a trajectory sampled from the user policy. The user policy is evaluated and the right image shows the states visited for the learned policy within the risk bounds.

We only learn the user policy for trajectories lying within our risk bounds, which ensures that we are still learning human's preference but not incorporating unreasonable preferences. We then plot a heat map of the state visitation frequencies for running the learned user's policy. It shows that our evaluation is able to avoid bad states by using this approach - the learned policy avoids the high-risk path that circles the -3 reward state.. Figure 5 shows this result.

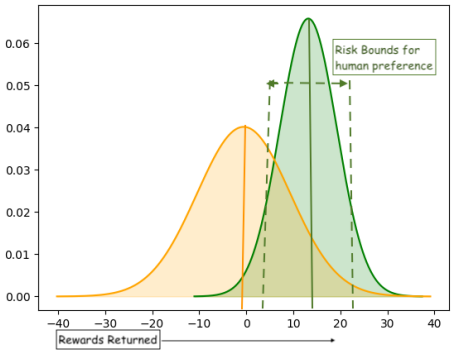

Figure 5.2: Evaluation of Case-1 in the grid game

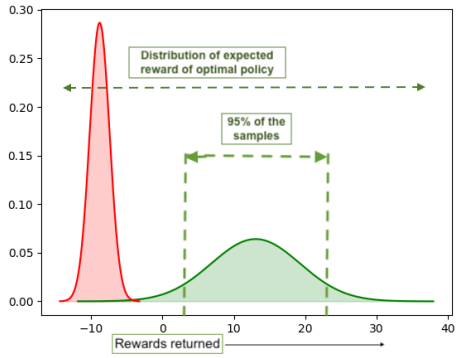

Figure 5.3: Evaluation of Case-2 in the grid game

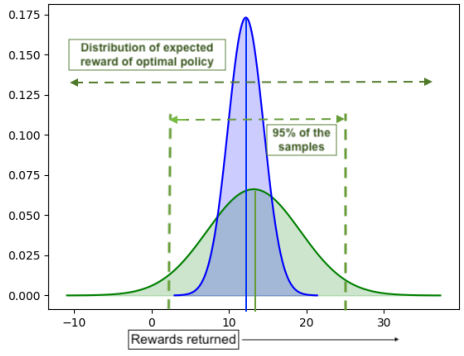

Figure 5.4: Evaluation of Case-3 in the grid game

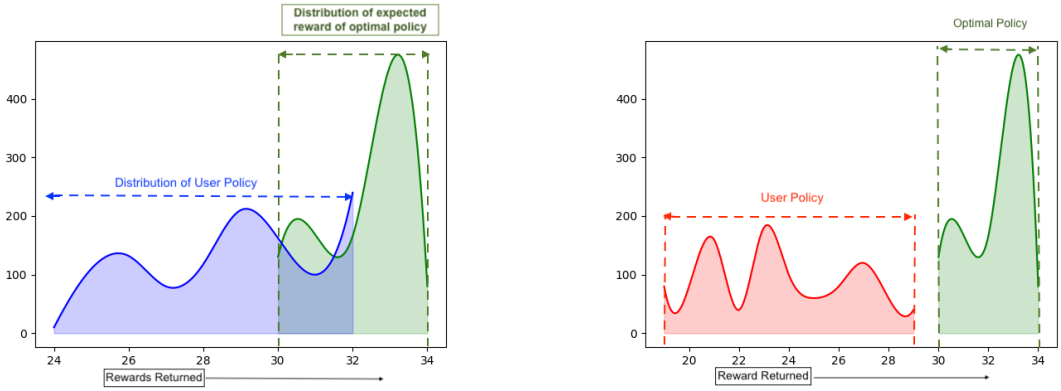

Figure 5.5: Comparison of different user policies with the optimal policy in the driving simulation

We get similar results for the 'highway-env'. Analysing cases of two different driver policies 1) an aggressive driver who is able to complete 30 time steps without crashing and 2) a driver who ends up crashing, we find that the evaluation is able to remove demonstrations exhibiting the collisions from the learned user policy. It also shows that the first user's policy is not acceptable in the whole range of returns, only part of it lies within the bounds of our risk. Learning a policy from the trajectories that lie only within the risk bounds, we see that the learned policy does not exhibit risky behaviour.

Chapter 6

Conclusions

In this work, we build upon a new definition of preference and identify it's role as a tie-breaker. We then introduce an evaluation criteria that is consistent with the given definition of preference. Comparing a user's policy to the optimal policy in an environment provides a strong initial evaluation for learning human preferences safely.

The approach shows promising results in learning user preferences while avoiding risky situations. We present experiments on two environments - a grid game and the OpenAI gym 'highway-env' and do a case by case analysis of different user policies in each environment. We apply our evaluation to each of the user policies and compare results. We see that the approach is to able to determine when human demonstrations exhibit unreasonable behaviour. The evaluation is also able to remove failed demonstrations from the learned policy.

Our work provides a novel evaluation method for learning human preferences using inverse reinforcement learning, while also avoiding any possible catastrophic outcomes of incorporating preference.

Chapter 7

Future Work

7.1 Designing threshold for risk

This work introduces a threshold for the maximum risk we can take in a given environment. The proposed approach to finding the bounds of risk are dependent on the distribution of rewards from roll outs of the optimal policy. This defines a very strict bound on our tolerance for risk. In the future, we are interesting in relaxing these bounds and exploring other designs for the threshold that can incorporate other compatible polices, while still ensuring safety.

Our threshold for risk tolerance is based on the total return of trajectories. This limits our learning to complete trajectories. In the future, we are interested in including feature-level thresholds, which might help us define more structured bounds. This might also help us to learn from trajectory segments, as we will be able to evaluate each state for its risk.

7.2 Reward Design

Currently, we use hand-crafted features for learning reward function from user demonstrations. Hand-crafting rewards is challenging and might fail to capture all the important aspects of the task you wish to learn. For future work, it might be interesting to explore other approaches for learning the reward, for example learning features from raw states, which would be a form of supervised learning.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In ICML '04: Proceedings of the twenty-first international conference on Machine learning, page 1, New York, NY, USA, 2004. ACM.

[2] Pieter Abbeel and Andrew Y. Ng. Exploration and apprenticeship learning in reinforcement learning. New York, NY, USA, 2005. Association for Computing Machinery.

[3] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety, 2016.

[4] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. Robotics and Autonomous Systems, 57(5):469–483, 2009.

[5] Andrea Bajcsy, Dylan Losey, Marcia O'Malley, and Anca Dragan. Learning from physical human corrections, one feature at a time. 03 2018.

[6] Mark Beliaev, Andy Shih, Stefano Ermon, Dorsa Sadigh, and Ramtin Pedarsani. Imitation learning by estimating expertise of demonstrators, 2022.

[7] A. Billard and D. Grollman. Robot learning by demonstration. Scholarpedia, 8(12):3824, 2013. revision #138061.

[8] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. Robot Programming by Demonstration. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[9] Daniel S Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. April 2019.

[10] Benjamin Burchfiel, Carlo Tomasi, and Ronald E. Parr. Distance minimization for reward learning from scored trajectories. In AAAI Conference on Artificial Intelligence, 2016.

[11] Erdem Bıyık. Learning preferences for interactive autonomy, 2022.

[12] Erdem Bıyık, Aditi Talati, and Dorsa Sadigh. Aprel: A library for active preference-based reward learning algorithms, 2022.

[13] Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023.

[14] Layla El Asri, Bilal Piot, Matthieu Geist, Romain Laroche, and Olivier Pietquin. Score-based inverse reinforcement learning. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[15] Daniel H Grollman and Aude Billard. Donut as i do: Learning from failed demonstrations. In 2011 IEEE International Conference on Robotics and Automation, pages 3804–3809, 2011.

[16] Sydney M. Katz, Amir Maleki, Erdem Bıyık, and Mykel J. Kochenderfer. Preference-based learning of reward function features, 2021.

[17] W. Bradley Knox, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and Peter Stone. Reward (mis)design for autonomous driving, 2022.

[18] W. Bradley Knox, Stephane Hatgis-Kessell, Serena Booth, Scott Niekum, Peter Stone, and Alessandro Allievi. Models of human preference for learning reward functions, 2022.

[19] Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training, 2021.

[20] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019.

[21] Siyuan Liu, Miguel Araujo, Emma Brunskill, Rosaldo Rossetti, Joao Barros, and Ramayya Krishnan. Understanding sequential decisions via inverse reinforcement learning. In 2013 IEEE 14th International Conference on Mobile Data Management, volume 1, pages 177–186, 2013.

[22] Lev McKinney, Yawen Duan, David Krueger, and Adam Gleave. On the fragility of learned reward functions, 2023.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[24] Andrew Ng and Stuart Russell. Algorithms for inverse reinforcement learning. ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning, 05 2000.

[25] Stefanos Nikolaidis, Ramya Ramakrishnan, Keren Gu, and Julie Shah. Efficient model learning from joint-action demonstrations for human-robot collaborative tasks. New York, NY, USA, 2015. Association for Computing Machinery.

[26] Malayandi Palan, Nicholas C. Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning reward functions by integrating human demonstrations and preferences, 2019.

[27] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research, 22(268):1–8, 2021.

[28] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. pages 2586–2591, 01 2007.

[29] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. Maximum margin planning. New York, NY, USA, 2006. Association for Computing Machinery.

[30] Dorsa Sadigh, Anca Dragan, Shankar Sastry, and Sanjit Seshia. Active preference-based learning of reward functions. July 2017.

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[32] Kyriacos Shiarlis, Joao Messias, and Shimon Whiteson. Inverse reinforcement learning from failure. 2016.

[33] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018.

[34] Aaquib Tabrez, Shivendra Agrawal, and Bradley Hayes. Explanation-based reward coaching to improve human performance via reinforcement learning. In 2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 249–257, 2019.

[35] Xiaofei Wang, Kimin Lee, Kourosh Hakhamaneshi, Pieter Abbeel, and Michael Laskin. Skill preferences: Learning to extract and execute robotic skills from human feedback, 2021.

[36] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay, 2017.

[37] Aaron Wilson, Alan Fern, and Prasad Tadepalli. A bayesian approach for policy learning from trajectory preference queries. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 25. Curran Associates, Inc., 2012.

[38] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning, 2016.

[39] Bradley Hayes Xinyu Cao. "preference in irl" - what does it mean?

[40] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In AAAI Conference on Artificial Intelligence, 2008.