Decreasing the number of demonstrations required for Inverse Reinforcement Learning by integrating human feedback

Zanyar Ogurlu[1]
Supervisor(s): Luciano Cavalcante Siebert[1]**, Antonio Mone**[1]
[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2024

Name of the student: Zanyar Ogurlu
Final project course: CSE3000 Research Project
Thesis committee: Luciano Cavalcante Siebert, Antonio Mone

Abstract

The main concept behind reinforcement learning is that an agent takes certain actions and is rewarded or punished for these actions. However, the rewards that are involved when performing a certain task can be quite complicated in real life and the contribution of different factors in the reward function is often unknown. From this problem emerges reward learning, which is the process of learning the reward function of an environment. There are several techniques for performing reward learning. We can view these different techniques within 2 different high-level categories: Learning from demonstrations and learning from feedback. IRL (Inverse Reinforcement Learning) is a way of learning from demonstrations. Meanwhile, RLHF (Reinforcement Learning from Human Feedback) is a way of learning from feedback.

In this paper, we are proposing the approach of training a reward learning agent, first with IRL and then with RLHF. IRL provides the benefit of learning a reward function quite quickly, however, it can suffer from the presence of sub-optimal demonstrations from the expert. Meanwhile, RLHF is slower at learning the reward function from scratch. Hence, we are proposing an approach where we integrate RLHF as a way to fine-tune the initial reward function calculated by IRL. By doing so, we are aiming to alleviate the negative effect of sub-optimal expert demonstrations on IRL.

We test and evaluate our methodology on the cart pole environment from the *seals* library. We compare the results from our approach to reward learning from only expert demonstrations, without integrating human feedback (i.e. only IRL). The obtained results suggest that, RLHF might in fact not be a good complement for IRL, specifically when we have sub-optimal expert demonstrations. In fact, we found that applying RLHF on top of IRL can even drop the performance of the resulting reward function, which challenges our initial hypothesis regarding the complementarity between these two methods.

1 Introduction

Learning by experience is quite a natural and significant phenomenon in humans' lives. As an example, let's take a child who is learning how to walk. The child does not really have prior information about what it is like to walk. Instead, they try certain movements and see whether those movements lead to success or failure. In case of success, they are rewarded by their parents, and in case of failure they fall or wobble. This is indeed a very simple example of a reinforcement process, and is at the basis of reinforcement learning. Reinforcement learning is a machine learning paradigm that focuses on teaching an agent how to optimally make sequential decisions within a certain context [18]. Figure 1 represents the logic behind a standard reinforcement learning approach. In this figure, we see two main elements: The agent and the environment. The environment defines the transition function and reward function (which is known in a standard RL setting), while the agent has a certain policy, or in other words, a course of action on the environment.

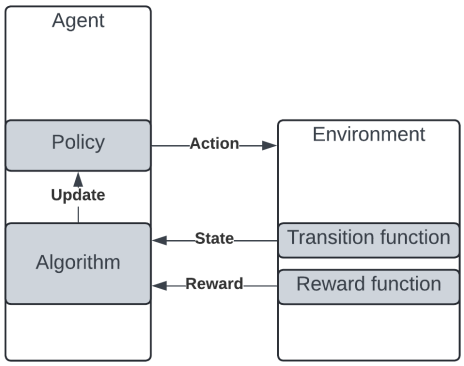


Figure 1: Intuition behind reinforcement learning, inspired by papers [16] and [9]

In reality, we are often missing a very important piece of the puzzle - the reward function. Though it might be somewhat easy to build a reward function manually in simple environments such as board games, often times the reward function can not be hard-coded so accurately. In these cases, we benefit from having an agent learn the reward function instead of having it pre-built.

Inverse reinforcement learning (IRL) is a way of learning the reward function from provided expert demonstrations [1] (or in some cases the expert policy directly). However, there is one issue in this method, which can simply be described as being overly reliant on the provided expert demonstrations [5]. Oftentimes IRL algorithms assume that the expert demonstrations are carried out perfectly and without error, which can be a false assumption in many cases. Additionally, IRL algorithms can require many demonstrations in order to yield good results, which can be difficult to gather. Lack of sufficiently optimal, or sufficiently many expert demonstrations reduces the performance of IRL.

As a way to address this issue, we are proposing to incorporate another learning technique called Reinforcement Learning from Human Feedback (RLHF) to complement IRL. In other words, we are proposing an approach where we first perform reward learning via IRL, and then via RLHF as a way to fine-tune the reward function. We aim to investigate whether RLHF is a natural complement for IRL and can reduce the over-reliance on expert demonstrations.

1.1 Related Work

In previous literature, the most similar approach to ours was proposed by Ezzeddine et al. [5]. In this paper they are using max-entropy IRL to train the agent at the beginning, and then apply RLHF on top of that. The main difference between our research and theirs is that they are using RLHF to particularly increase reward learning performance. However, what we are mainly investigating is whether we can decrease the number of expert demonstrations needed for IRL by using RLHF.

Another similar approach to ours was proposed by Kunapuli et al. [10]. In this work, they are improving IRL with expert advice, which includes reducing the number of demonstrations needed. The advice is being provided in the form of state, action and reward

preferences. What mainly separates our work from theirs is the algorithms we are choosing to incorporate. The kind of feedback we are using for the RLHF part is binary feedback between two trajectories, which is quite different from theirs.

Furthermore, El Asri et al. [4] proposes a similar framework to ours, however instead of using binary evaluation between two trajectories, they are using a scoring mechanism where the human expert assigns a score to the trajectories. Their proposed methodology seems to be good at dealing with sub-optimal expert demonstrations, which is something we are also aiming to address. However, this paper also does not focus directly on decreasing the number of demonstrations that IRL needs, so we could say that the goal of this research is somewhat different from ours.

2 Research Question

The main research question that will be addressed in this paper is:

To what extent can RLHF complement IRL to reduce the number of expert demonstrations needed for IRL?

While looking into this question, there are some other sub-questions that will also be addressed, which are:

1. *How can expert feedback be incorporated into a system that uses IRL (specifically Adversarial IRL) and what kind of advantages does this approach bring?*
2. *In which circumstance does adding RLHF to a system that uses IRL not benefit the performance of the system?*

3 Preliminaries

Inverse Reinforcement Learning (IRL) and Reinforcement Learning from Human Feedback (RLHF) are both relatively recent fields in machine learning. This section provides a background into the theory of RL, IRL and RLHF, introduces the fundamental concepts within these fields that are relevant to this research paper.

3.1 Reinforcement Learning (RL)

We can use a Markov Decision Process to model a reinforcement learning problem [13]. A Markov Decision Process can be formalized as a tuple (S, A, R, P, γ). Below is a brief explanation of these 5 different elements that define a Markov Decision Process [8]:

- S is the set of states that an agent can be in.
- A is the set of actions that an agent can perform.
- R is the reward function, which defines the reward obtained by an agent when it performs a certain action in a certain state.
- P is the transition function, which describes the probability that the agent will move from state s to s' by performing action a.

- γ is the discount factor, which defines how much the agent prefers earlier rewards to later ones.

Another important term that is used in reinforcement learning theory is the policy, which can be defined as a function that models how likely an agent is to perform a particular action at a given state. Traditional reinforcement learning attempts to obtain an optimal policy, given a particular Markov Decision Process.

3.2 Inverse Reinforcement Learning (IRL)

The concept of IRL was first proposed in 1998 by Stuart Russell. In his paper [14], Russell reasons for the significance of IRL via making connections with how learning occurs in nature. He states that in nature, the reward function is often unknown. For instance, when a person is running, there are many attributes that might be seen as part of the reward function, such as the speed, efficiency, etc. It is not very intuitive that all these attributes are weighted and combined by the person a priori. It seems much more realistic that the contribution of these different attributes to the reward is learned by the human, which in a way, is an inverse reinforcement learning problem. This analogy from nature can also be applied to virtual agents, oftentimes the reward function is not known and has to be learned. Figure 2 shows the intuition behind the standard IRL setting - inspired by [15]. The essential idea here is that an IRL algorithm aims to calculate a reward function that explains expert trajectories (demonstrations) and thereby obtain an accurate model for the actual reward function of the environment.

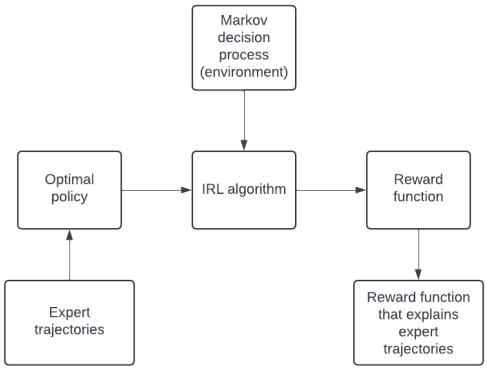


Figure 2: Intuition behind IRL, inspired by [15]

There are several categories of Inverse Reinforcement Learning techniques. The most common ones are max-margin methods, Bayesian methods and maximum entropy methods. The max-margin methods maximize the margin between the optimal policy and all other possible policies. These methods often allow for only simple kinds of modelling. The Bayesian methods, on the other hand, maximize the posterior distribution of the reward, which allows for more complex kinds of modelling.

Often, with max-margin and Bayesian methods, it is hard to obtain the best policy. These methods focus on finding a policy that matches the provided demonstrations. However, there

are many possible policies fitting the demonstrations, and the one that is found is likely not the correct one. This is especially a problem when the provided expert demonstrations are sub-optimal, because in that case, the agent will basically be over-fitting the reward function on inaccurate demonstrations. By using maximum entropy IRL algorithms, we are able to choose the best policy over the alternatives and hence arrive at a single stochastic policy [19].

Even though maximum entropy IRL algorithms are good at choosing the optimal policy for a given set of demonstrations, they still fall short at finding rewards that best explain the given policy. One of the algorithms that can be used to address this issue is AIRL (Adversarial Inverse Reinforcement Learning). The AIRL algorithm works similar to Generative Adversarial Imitation Learning. What makes it different is its emphasis on learning rewards that are invariant to changing dynamics in the environment [6]. Due to these reasons, we will be using the AIRL algorithm as the IRL component in our experiment setup.

3.3 Reinforcement Learning from Human Feedback (RLHF)

RLHF is a generalization made from a prior term that was in use, which is PbRL (Preference-based Reinforcement Learning). While PbRL focuses on finding the objective policy from the provided evaluative feedback, RLHF has more of an emphasis on reward learning and is less restricted in terms of the type of feedback that is integrated [9]. In this way, RLHF can be seen as a very natural complement for IRL, since they both share the objective of reward learning as opposed to direct policy optimization. Figure 3 shows the general logic behind RLHF algorithms. Basically, we are using a trainer to give feedback and we use that feedback to update the reward function, simultaneously performing standard RL to learn the optimal policy.

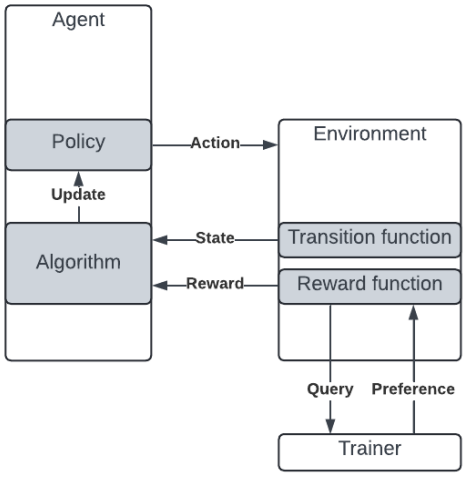


Figure 3: Intuition behind RLHF, inspired by [9]

There are several types of feedback that can be used for RLHF (such as numerical, binary evaluative). However, in this research we will be using preference comparisons RLHF, where the trainer will provide a comparison between two trajectory segments (i.e. which one is

better). This is the standard RLHF implementation from the *imitation* library. It has been shown to be able to solve complex RL tasks on a variety of environments without access to the reward function [3], which makes it suitable for our purposes.

4 Methodology

When integrating IRL and RLHF, it is most sensible to first learn an intermediate policy and reward function via IRL and then apply RLHF to fine-tune. This is because IRL can provide us with a quicker learning at the beginning while learning from feedback often has a slower convergence rate when it starts from scratch [5]. Figure 4 shows a high-level visualization of the methodology we use in our experiments.

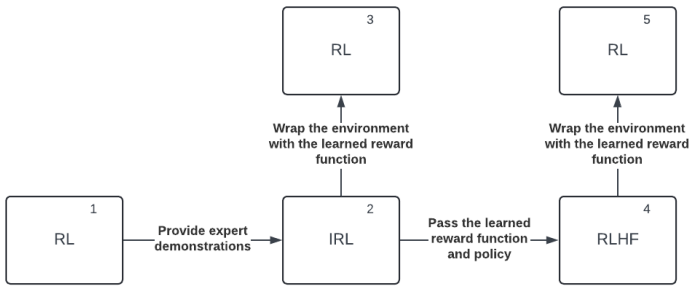


Figure 4: Methodology of our experiment with 3 main components

Described below is a more detailed step by step description of the methodology by which we conduct our experiments. You can also access our implementation of this methodology via our public GitHub repository.

1. **RL**

   1.1. Loading an environment, which is a representation of the Markov Decision Process that will be used in the experiment. This environment will have a known and simple reward function.

   1.2. Training an agent using RL on the environment that is loaded. The training will be made to varying levels of completion. As a result, some trained models will have optimal behavior, while some other models will have sub-optimal behavior.

2. **IRL**

   2.1. Generating demonstrations based on the RL model that was trained in the previous step. With some models these demonstrations will be optimal, and with others sub-optimal. The purpose of this is to see how IRL is affected by the sub-optimality of the provided demonstrations, and the extent to which we can complement this via the use of RLHF.

   2.2. Initializing an agent that is "blind" or in other words, does not know the reward function for the environment.

2.3. Training the agent from the previous step using AIRL to learn an initial reward function, saving the resulting policy and reward function.

2.4. Performing regular RL on the initial learned reward function, and evaluating it against the original environment with the actual reward function.

3. **RLHF**

3.1. Generating trajectories, using the agent from the IRL part. This means the initial trajectory generation is not random but will be based on the policy trained by the IRL algorithm.

3.2. Feeding the reward function that was discovered by the agent in the IRL stage to the RLHF algorithm.

3.3. Training the agent via RLHF, saving the resulting policy and reward function.

3.4. Performing regular RL on the final learned reward function, and evaluating it against the original environment with the actual reward function.

5 Experimental Setup and Results

The first stage in the experimental setup is loading an environment, or in other words, a Markov Decision Process. There are quite a few environments readily implemented in the *gymnasium* [17] and *seals* [7] libraries. In this experiment, an environment with a simple reward function will be used. Given the computational power that is available to us at this point, we believe this to be a sensible choice, as this experiment will perform RL, IRL and RLHF training all at once. Also, performing the experiment on a relatively simple environment will yield mostly deterministic results, which will increase the reproducibility of our experiment. Henceforth, in this experiment, we will be using the cart pole environment from the *seals* library.

In the cart pole environment, there is a a rigid pole which is hinged to a cart. The cart is free to move within the bounds of a one-dimensional track. The pole is free to move only in the vertical plane of the cart and the track. The controller can apply a "left" or "right" force of fixed magnitude to the cart at discrete time intervals [2]. The observation space (states) are defined by 4 elements: Cart position, cart velocity, pole angle and pole angular velocity. Figure 5 shows the visual representation of the cart pole environment.

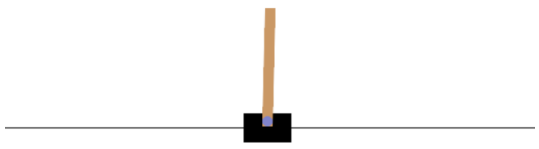


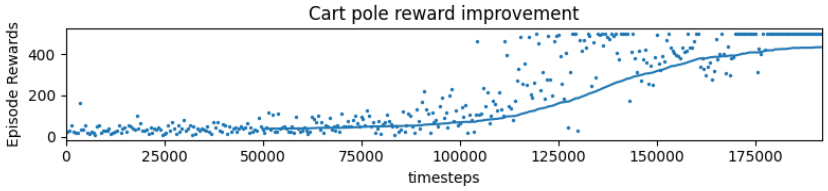
Figure 5: Cart pole environment from the *seals* library

The first step in the experiment is to set up the trainer agent that will be generating the demonstrations for the IRL part, which is the initial RL stage described in our methodology.

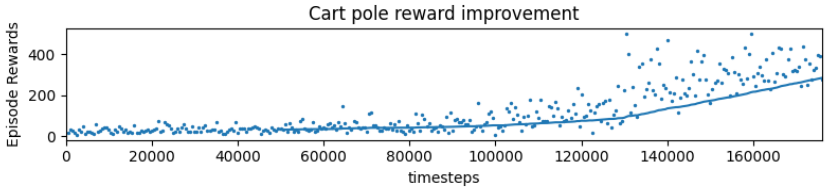
In other words, we need to train the expert. For this, we will be using the Deep Q-Network (DQN) implementation from the *Stable Baselines3* [12] library. This implementation is an instance of deep reinforcement learning, which is an improvement on standard reinforcement learning and exceeds it in terms of successfully learning control policies [11].

We will be including both optimal and sub-optimal expert demonstrations during the IRL training. Hence, we need to train expert models with a varying degree of optimality in their behavior by the end of the training process. This way, we can also study the effect that the optimality of the expert demonstrations has on the IRL process.

The graph in figure 6 shows the progression of the mean reward values per episode during the policy learning process. In total, we trained 2 different models using DQN. The first model has an optimal policy at the end of the training, in other words, the policy consistently receives the highest possible reward on the environment. The second model has a sub-optimal policy at the end of the training, however, it is still expected to perform decently. In other words, this second policy receives high rewards on average, but with high standard deviation as well.



(a) Nearly optimal expert



(b) Sub-optimal expert

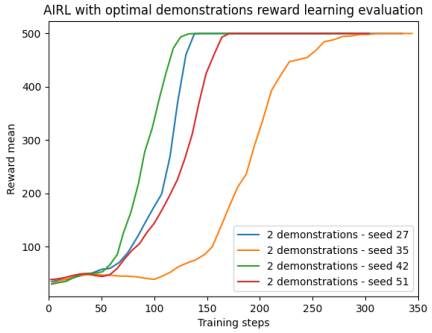
Figure 6: Different levels of sub-optimality while training AIRL

5.1 Reward learning via AIRL

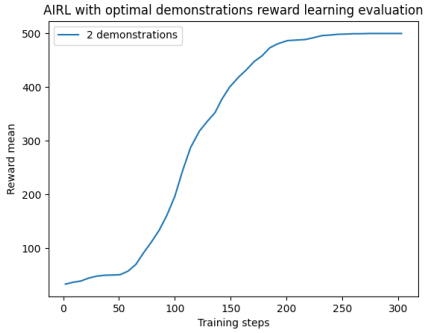
We firstly performed reward learning with AIRL alone. For this stage, we use the following process: Using AIRL for reward learning, and after that, feeding the learned reward function to the environment and performing regular RL. Then we will evaluate the policy that results from this process against the actual environment with the original reward function. If the learned policy does not perform well enough on the actual environment, this will inform us that AIRL was not very good at learning the true reward function.

We started by running the AIRL algorithm with perfectly optimal expert demonstra-

tions, without any divergence from the highest reward obtainable per episode. By using such an expert, the AIRL algorithm was able to learn the reward function perfectly, as expected. Figure 7 the reinforcement learning process resulting from the learned rewards.



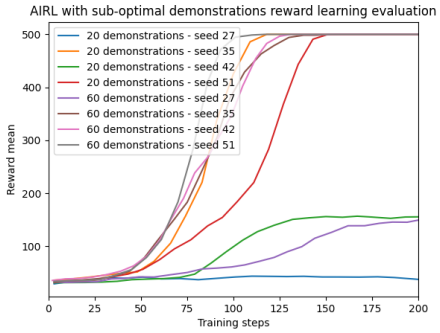
(a) Reinforcement learning using AIRL rewards provided optimal expert demonstrations with seeds of 27, 35, 42 and 51



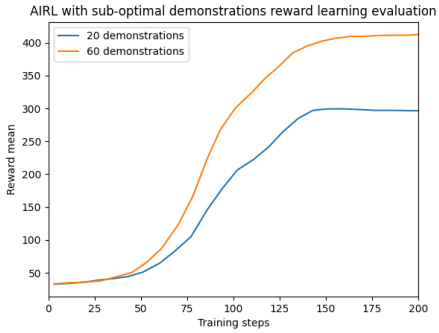
(b) Reinforcement learning using AIRL rewards provided optimal expert demonstrations averaged over different seeds

Figure 7: Reinforcement learning using AIRL rewards on the cartpole environment with 2 optimal demonstrations

As we see from figure 7, by using as little as 2 demonstrations, AIRL performs quite well, given that the demonstrations are optimal. However, as the optimality of the provided demonstrations drops, we see that the performance of AIRL will also drop. To test this, we provided the AIRL algorithm demonstrations from an expert that receives 474 reward per episode with a standard deviation of 64. Afterwards, we performed reinforcement learning on the learned reward function, which showed a progression as seen in figure 8.



(a) Reinforcement learning using AIRL rewards provided sub-optimal demonstrations with seeds of 27, 35, 42 and 51



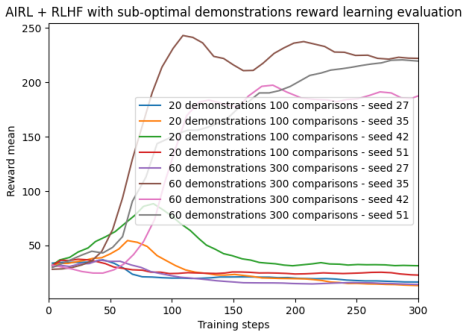
(b) Reinforcement learning using AIRL rewards provided sub-optimal demonstrations averaged over different seeds

Figure 8: Reinforcement learning using AIRL rewards on the cartpole environment using 20 and 60 noisy/sub-optimal demonstrations with a 474 reward mean and 64 standard deviation

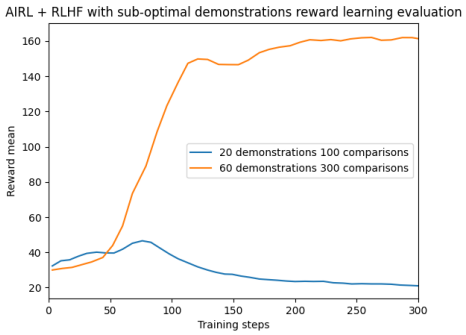
Indeed, we see that when the expert demonstrations are sub-optimal, reward learning via AIRL becomes much less sufficient, especially as we decrease the number of demonstrations provided to the agent.

5.2 Proposed Approach: Reward learning via AIRL and RLHF

Firstly, we used the same sub-optimal expert demonstrations that we used in the previous stage, with a reward mean of 474 and standard deviation of 64. As we did with the previous part in our experiment, we trained the AIRL component of our algorithm with 20 and 60 expert demonstrations. Afterwards, we passed the policy and reward function learned by the AIRL component to the RLHF component, and performed a preference comparisons training with respectively 100 and 300 comparisons. Finally, we wrapped the cartpole environment with the reward function yielded by the RLHF algorithm, and we performed RL on this environment. Figure 9 shows the mean episode rewards obtained during this training process.



(a) Reinforcement learning using AIRL and RLHF rewards provided sub-optimal demonstrations with seeds of 27, 35, 42 and 51



(b) Reinforcement learning using AIRL and RLHF rewards provided sub-optimal demonstrations averaged over different seeds

Figure 9: Reinforcement learning using AIRL and RLHF rewards on the cartpole environment using 20 and 60 noisy/sub-optimal demonstrations with a 474 reward mean and 64 standard deviation

We see from figure 9 that, even at the end of this RL training process, we obtained episode rewards of only 160 on average. This informs us that the reward function that resulted from performing an RLHF training on top of an AIRL training is not an accurate or effective model on the actual reward function for the environment. In fact, by making comparisons with the results obtained in the previous stage of our experiments, we see that AIRL by itself seems to be a relatively better option. Hence, we can conclude that in the setting which we used, RLHF does not form a good complement with AIRL, and does not help decrease the required number of expert demonstrations.

6 Responsible Research

Throughout the research process, there has been many measures taken in order to ensure that the results are reproducible, and can be verified and used by other researchers in the future.

In order for the experiment to be reproducible, the libraries and environments that are being used have been explained in detail. Additionally, all the code that has been implemented while running the experiments are placed in a public repository so that it can be accessed and easily. By looking at our code, it can be verified that our intended methodology has been implemented accurately. The code includes proper structure so as to not confuse the reader, and the purpose of key files and directories are outlined in the README markdown.

Aside from the Python code, we have also made available the expert models that were generated in preparation for the IRL component. This is because it can be hard to generate the exact same expert models even if one knows the number of training steps used and other hyper-parameters. All the models that have been generated are placed in the repository, and the naming indicates the policy evaluation results that were obtained from these models. We believe that these models are important for the purpose of making our experiment completely reproducible.

Our research does not use any human participants, or involve humans at any point during its course. All the experiments are performed on a simple virtual environment, and the actions are all performed by a virtual agent. This way, we are ensuring that no one is being affected by our experiments, be it something simple like taking people's time. In addition, the environment we are using is in no way related to a real-life scenario. Hence, we do not expect our research to have any sort of relation to issues that may be considered sensitive.

7 Discussion

We started by showing that IRL does not perform well on its own when the expert demonstrations provided to it are sub-optimal. Furthermore, in that kind of setting, in order to increase the performance of the IRL model, we need to use more expert demonstrations. However, this is not always feasible, because in more complicated environments it can be quite time and effort consuming to collect expert demonstrations. This is where we thought RLHF could come into play. Our expectation was that we could use RLHF as a means to fine-tune the reward function obtained by IRL as a basis, and this way we could decrease the number of demonstrations needed by IRL - especially in the case where we have sub-optimal demonstrations. For this, we performed experiments on the cart pole environment from the *seals* library, to see whether RLHF would complement IRL (specifically Adversarial IRL).

We verified that indeed, IRL works very well if it is provided optimal demonstrations. Additionally, since the demonstrations are optimal, the algorithm does not require many of them. This is because more demonstrations does not bring any new information to the model if they are all optimal, especially in the basic cart pole environment we chose to use where the goal of the agent and the reward structure are straight-forward. However, if we were working with a multi-objective model, this would most likely not be the case, and the number of demonstrations would have more effect on the IRL training, even if the demonstrations are optimal. This can be considered a limitation of our research.

Furthermore, we performed experiments in the setting with IRL and RLHF together. We saw that RLHF does not complement IRL meaningfully when the expert demonstrations provided to IRL are sub-optimal. In fact, when we apply RLHF, the reward function becomes worse compared to the reward function obtained after applying IRL. The reason for this is unknown at this point, but we hypothesize that IRL learns a reward function that is far from the actual reward function for the chosen environment. When we apply RLHF later on, the

"false" reward function learned by the IRL component misleads the preference comparisons algorithm. This hypothesis needs to be studied and backed up by further research.

To elaborate further upon the limitations of our research, the experiments were performed on the basic cart pole environment. This was mostly caused by a lack of time and resources. Furthermore, our intention with this research was not to prove the complementarity or non-complementarity between IRL and RLHF, but rather was to investigate whether they can work together on a simple model. Additionally, we only looked into a certain IRL algorithm, being AIRL, and a certain RLHF algorithm, being preference comparisons. This can also be seen as a limitation, since it is likely that different types of IRL and RLHF algorithms can complement each other better.

8 Conclusions and Future Work

In this paper, we have looked into the complementarity between Inverse Reinforcement Learning (IRL) and Reinforcement Learning from Human Feedback (RLHF). Specifically, we attempted to complement these two methods in order to decrease the number of demonstrations needed for the IRL component. For this, we proposed an approach where we train an agent (policy) and a reward network, initially via Adversarial IRL, and then preference comparisons RLHF. At the end, we evaluated the learned reward network by comparing it to that which is learned by Adversarial IRL alone.

By training an agent via only AIRL, we verified that it indeed performs well in case the provided expert demonstrations are optimal. We also verified that when the demonstrations are sub-optimal, AIRL does not perform as well, especially when we provide it with a smaller number of demonstrations.

We need the complementarity between IRL and RLHF to work specifically when the expert demonstrations are sub-optimal. However, our experiments showed that, when the demonstrations are sub-optimal, RLHF does not complement IRL, in fact it results in a worse reward function at the end. Hence, we were not able to use this approach to decrease the number of demonstrations needed for IRL.

In the future, we are planning to run our experiment with other environments and using other types of IRL and RLHF algorithms. One reason why we think RLHF did not complement IRL that well in our experiment is that, we ran the two algorithms, one after the other. If we were to instead develop a broader algorithm that combines IRL and RLHF under the same process more naturally, we could possibly yield better results. Another potential approach is to revert the ordering of IRL and RLHF, which we theoretically do not expect to perform better than our approach, but still find to be worth researching further.

References

[1] Stephen Adams, Tyler Cody, and Peter A. Beling. A survey of inverse reinforcement learning. *Artificial Intelligence Review*, 55(6):4307â4346, Feb 2022.

[2] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983.

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

[4] Layla El Asri, Bilal Piot, Matthieu Geist, Romain Laroche, and Olivier Pietquin. Score-based inverse reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016)*, Singapore, Singapore, May 2016.

[5] Ali Ezzeddine, Nafee Mourad, Babak Nadjar Araabi, and Majid Nili Ahmadabadi. Combination of learning from non-optimal demonstrations and feedbacks using inverse reinforcement learning and bayesian policy improvement. *Expert Systems with Applications*, 112:331–341, 2018.

[6] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning, 2018.

[7] Adam Gleave, Pedro Freire, Steven Wang, and Sam Toyer. seals: Suite of environments for algorithms that learn specifications. `https://github.com/HumanCompatibleAI/seals`, 2020.

[8] Michael Hu. Markov decision processes. *The Art of Reinforcement Learning*, page 21â46, 2023.

[9] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hullermeier. *A survey of reinforcement learning from human feedback*, Dec 2023.

[10] Gautam Kunapuli, Phillip Odom, Jude W. Shavlik, and Sriraam Natarajan. Guiding autonomous agents to better behaviors through human advice. *2013 IEEE 13th International Conference on Data Mining*, Dec 2013.

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

[12] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3. `https://github.com/DLR-RM/stable-baselines3`, 2019.

[13] Rafael Ris-Ala. *Fundamentals of Reinforcement Learning.* Springer International Publishing AG, 2023.

[14] Stuart Russell. Learning agents for uncertain environments (extended abstract). *Proceedings of the eleventh annual conference on Computational learning theory*, Jul 1998.

[15] Syed Ihtesham Shah and Giuseppe De Pietro. An overview of inverse reinforcement learning techniques. *Intelligent Environments 2021*, Jun 2021.

[16] Ashish Kumar Shakya, Gopinatha Pillai, and Sohom Chakrabarty. Reinforcement learning algorithms: A brief survey. *Expert Systems with Applications*, 231:120495, Nov 2023.

[17] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel GoulÃ£o, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea PierrÃ©, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023.

[18] Phil Winder. *Reinforcement learning.* 2020.

[19] Brian Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. *Maximum Entropy Inverse Reinforcement Learning.* AAAI Press, 2008.