Analysis Of T-REX Inverse Reinforcement Learning When Provided Natural Data

Zac C. Penson

Received August 13, 2024
Accepted September 23, 2024
Electronic access October 15, 2024

Background: Reinforcement learning models have long utilized machine-based trial and error to achieve a programmed task. This method of learning may not replicate the complexities of real-world learning. These limitations in reinforcement learning have spurred the development of inverse reinforcement learning (IRL). Prior data has shown the utility of IRL in Trajectory-ranked Reward Extrapolation (T-REX) when providing a set of criteria of poor demonstrations. Our work differs in that we assess the ability of IRL without pre-determined demonstrations, using MiniGrid to test the ability of T-REX when given natural data demonstrations.
Methods: Three different environments — Empty, DoorKey, Dynamic Obstacles — were utilized to assess if an agent could correctly navigate to an endpoint. Additionally, different quantities of demonstrations were provided in each environment, either 4, 6, 8 or 10. Successful returns were quantified by the agent attaining this goal.
Results: In the Empty environment, agent success improved as the number of demonstrations increased. Although yield levels were lower in the more stressful DoorKey and Dynamic Obstacle environments, agent success still improved as increasing demonstrations were provided. In these environments the agent often failed to reach the end zone when provided with lower numbers of demonstrations.
Conclusion: The results supported our hypothesis that when provided with natural data, the success rate of the agent would directly correlate to the number of demonstrations provided. Future research should evaluate adding additional demonstrations and greater computer power to assess if agent learning improves.

Keywords: Machine Learning; Artificial Intelligence; Inverse Reinforcement Learning; Grid World

Introduction

One of our greatest human qualities is the ability to learn through trial and error. With the advent of Artificial Intelligence (AI), we have strived to teach computers a similar learning process. Learning a programmed task through random computer-generated trial and error is known as reinforcement learning. One way to demonstrate and simulate such an environment is by using a grid world. In a grid world, the agent, the figure navigating the environment, is tasked to make it to an endpoint. With reinforcement learning, the agent will use trial and error to make it to the end location by itself and then return the result to the user. When the agent reaches the end location, it is given a reward, usually a +1. This trial and error policy allows the agent to make an optimal route to reach the end location. Autonomous cars, gaming, CT scanners, and natural language processing are all examples of how reinforcement learning is used in everyday life[1]. However, traditional reinforcement learning methods often require simplification, and may not replicate the intricacies of real-world learning. This limitation has spurred the development of inverse reinforcement learning (IRL), where systems learn from observed behaviors, known as demonstrations, rather than trials alone[2]. In IRL, the agents can make more complex, human-like decisions by extracting the underlying reward function from observed demonstrations[3]. Similar to reinforcement learning, IRL can also be tested and applied in finite and grid spaces[4]. Understanding economic market dynamics, financial crime detection, and enhanced robotic movement are all current real-world applications of IRL[5]. Original IRL studies prioritized poor demonstrations in order to show the ability of the agent to improve upon the poor performances[3]. While this is useful information for the groundwork of IRL, it is not practical to how true human learning occurs. In this manner, humans would have to create demonstrations that meet a standard of imperfection for this type of IRL learning to occur. Our study examines how IRL learning works in a more organic fashion. We utilized a reward-learning-from-observation algorithm called T-REX that was introduced by Brown, et al[6]. T-REX attempts to utilize IRL to extrapolate beyond a set of ranked demonstrations in order to infer a higher quality reward function. Building off work by Ng, et al, we attempted to minimize IRL degeneracy by selecting a reward function that would clearly differentiate

the observed policy from other sub-optimal policies.[7] Using an existing program called Minigrid, we ran different demonstrations to examine the effects on the T-REX algorithm and our agent.[8,9] Our work differs from Brown, et al, in that demonstrations were not prioritized, rather data was provided to the model in a more random manner (see more on the implications of "random" in the Collecting Demonstrations section). We were then able to assess the ability of the model to learn in a more human setting, where data is not pre-selected for T-REX. Similarly, Beliaev, et al, have worked to create the IRLEED algorithm, to perform without prior knowledge of demonstrator expertise and efficiently derive optimal policy from suboptimal demonstrations.[10] We hypothesized that when our agent is provided with natural data demonstrations, the agent's success would directly correlate to the number of demonstrations presented. We further hypothesized that IRL success would continue to scale in more complex MiniGrid environments when the number of demonstrations presented increased.

Methods

Collecting Demonstrations

As in the original T-REX paper, we use the term stages to indicate sets of demonstrations, where each subsequent stage includes all of the demonstrations of the prior stage[5]. For example, our first stage consisted of four unique demonstrations. These four demonstrations would be tested ten times each in order to get an average return for the first stage. Then for the following stage, two new demonstrations would be added on to the original four. These six demonstrations in stage 2 would then also be tested ten times to create another average return for the stage. Another two demonstrations would be added to the six prior demonstrations in stage 3, to make a stage with 8 total demonstrations. These were again tested ten times. Finally, an additional two demonstrations would be added to the eight prior demonstrations, to make a stage with 10 total demonstrations. Again, these were tested ten times to create an average for the last stage. This incremental addition of demonstrations allowed us to determine which stage was ideal for an optimal program. The process of generating demonstrations for each stage that were meaningful and conducive to the experiment was a vital component of our work. While humans cannot truly create "random" data demonstrations, our work attempted to diversify input demonstrations by providing differing levels of successful trials. Positive demonstrations were defined as those requiring less than 10 agent actions to complete the task. Moderate demonstrations consisted of 10 to 15 actions, while poor demonstrations required 16 or more actions by the agent. Approximately 45% of our demonstrations were positive outcomes, 35% moderate, and 20% negative. This dispersion of demonstrations allowed the potential for the IRL program to learn from its mistakes, while simultaneously having the opportunity to improve upon the positive demonstrations to attempt to produce an optimal solution.

Details

Software Requirements: Our software tool used in our investigation was MiniGrid, which offers a multitude of environments, with each being tunable for sizes and complexity[8]. In all of the environments, the agent is a red triangle, which has three available actions: rotate left, rotate right, and move forward. Each environment is situated along a 2D plane, which simplifies visualization and reduces complexity. The sparse representation of the 2D plane requires defining only two states, allowing the program to run faster and reducing the computational load on the device.

Experimental Setup and Design: In the experiment, we conducted trials in three different environments to test the behavior of the agent in various complex scenarios. Trials for the first 4 stages (10 runs per stage) were conducted in the Empty environment, where the red arrow was tasked to reach the green finish zone. Image 1 shows the progressive movement of the agent in the Empty environment, where the red agent moves forward on the grid and then turns right toward the green finish zone. In the Empty environment, the course was clear of obstacles, allowing multiple paths for the agent to reach the endpoint. The Empty environment uses the three main commands for the movements of the agent, which is presented in Table 1. These available keyboard commands are used to navigate the agent through the course in both the Empty and Dynamic Obstacles environments.

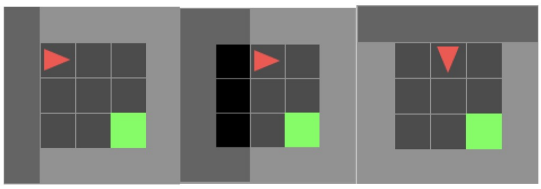

Fig. 1 Depiction of the progressive movement of the agent in the Empty environment.

Key	Name	Action
Left arrow	left	Turn left
Right arrow	right	Turn right
Up arrow	forward	Move forward

Table 1 The available keyboard commands used to navigate the agent through the course in the Empty and Dynamic Obstacles environments.

Our trials for the next 4 stages (10 runs per stage) were conducted in the DoorKey environment, in which the red arrow is tasked to reach the green finish zone (see Image 2). In this environment the agent must learn how to pick up a key (denoted

by a yellow "key" sprite), take it to a virtual door (denoted by a yellow square), open the door, and continue to reach the green finish zone to accumulate the point. Table 2 shows the five primary commands for the agent's movements and actions in the DoorKey environment.

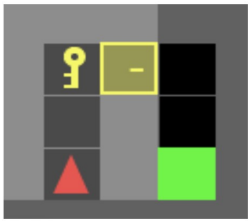

Fig. 2 The agent and its surroundings in the DoorKey environment. In this environment, the red agent is required to pick up the yellow "key" sprite and use it to open the virtual yellow door to reach the green finish zone.

Key	Name	Action
Left arrow	left	Turn left
Right arrow	right	Turn right
Up arrow	forward	Move forward
3	pickup	Pickup object
5	toggle	Open door with key

Table 2 The available keyboard commands used to navigate the agent through the course in the DoorKey environment.

Our trials for the final 4 stages (10 runs per stage) were conducted using the Dynamic Obstacles environment, in which the red arrow is tasked to reach the green finish zone (See image 3). In this environment, the red agent must avoid the blue moving obstacles (denoted by blue circles) to successfully reach the green finish zone and accumulate the point. The Dynamic Obstacles environment uses the same 3 main commands for the movements of the agent presented in table 1.

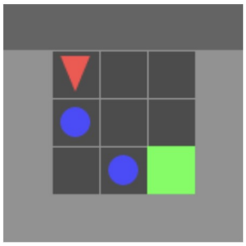

Fig. 3 The agent and its surroundings in the Dynamic Obstacles environment.

For each of the 3 environments, 40 total trials were performed (10 per demonstration stage). In all, four sets of a defined number of demonstrations were provided per environment. A demonstration is a human made process in which the agent is guided correctly (or incorrectly) to the end location, and humanly ranked based on success and efficiency. The demonstrations used for each environment were selected as described above in Collecting Demonstrations. Some were more efficient, some were less efficient. Not all demonstrations were perfect, allowing for greater interpretation from the program. However, every demonstration made it to the finish zone, as MiniGrid requires every demonstration to make it to the end before it can contribute to the overall run. The first stage of ten program runs would be given 4 demonstrations, the second would be given 6, the third would be given 8, and the fourth would be given 10 demonstrations. We would build upon our last set of demonstrations as we advanced. In other words, rather than moving from 4 distinct demonstrations to 6 entirely new ones, we would use the same four and add on 2 novel ones. This process was used for all of the environments. Demonstrations were ranked based on the number of actions, in which an action is defined by one of the following in either table 1 or 2.

Results

Success was measured in our data by the agent achieving its goal, and reaching the endpoint. MiniGrid returns 1 after a successful attempt, and 0 after a failure. A result of -1 is returned in Dynamic Obstacles if the agent collides with an object. A mean percentage of successful attempts was then calculated for each stage. 1 represents a 100% success rate of the agent. The closer to 1, the greater the percentage of times the agent reached the green finish zone at the end of the 6x6 MiniGrid environments. These mean percentages are presented for 4, 6, 8, and 10 demonstration stages throughout our trials per environment, and statistically significant differences between stages are reported. The initial 6x6 Empty Environment produced a return of 0.687 when 4 demonstrations were given. The Empty Environment produced a return of 0.847 when 6 demonstrations were given. The Empty Environment produced a return of 0.940 when 8 demonstrations were given. The Empty Environment produced a return of 0.947 when 10 demonstrations were given. The 8 and 10 demonstration groups produced statistically significant improvements in yield when compared to the 4 demonstration group, $p < 0.05$. It should be noted in the Empty Environment that runs #3 and #9 in the 4 demonstration trials had very poor success rates of 0.066 and 0.089, respectively. Also, Run #2 in the 6 demonstration trials failed to yield a positive result, returning a score of 0. The data is displayed below in Table 3.

The 6x6 DoorKey Environment produced a return of 0.010 when 4 demonstrations were given. The DoorKey environment produced a return of 0.103 when 6 demonstrations were provided. Only minimal improvement was observed when increasing to 8 demonstrations, producing a return of 0.140. Improvement in yield to 0.359 was observed in the DoorKey Environment when increasing to 10 demonstrations. The increase in the

# demos	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Run 6**	**Run 7**	**Run 8**	**Run 9**	**Run 10**	**Average Return**	**+ SD**
4	0.943	0.940	0.066	0.940	0.943	0.142	0.938	0.942	0.089	0.929	0.687	0.385
6	0.942	0.000	0.944	0.945	0.942	0.939	0.950	0.947	0.934	0.931	0.847	0.282
8	0.932	0.936	0.941	0.940	0.942	0.943	0.934	0.950	0.939	0.938	0.940	0.005
10	0.943	0.960	0.945	0.939	0.945	0.951	0.942	0.954	0.948	0.946	0.947	0.006

Table 3 Data from each run in the Empty environment (with the number of demonstrations provided to the agent). The highest mean success rate was observed in the stage that was provided with ten demonstrations.

# demos	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Run 6**	**Run 7**	**Run 8**	**Run 9**	**Run 10**	**Average Return**	**+ SD**
4	0.000	0.000	0.000	0.010	0.000	0.000	0.034	0.040	0.000	0.015	0.010	0.014
6	0.000	0.000	0.000	0.040	0.038	0.000	0.000	0.932	0.000	0.024	0.103	0.277
8	0.000	0.000	0.000	0.365	0.734	0.000	0.022	0.000	0.246	0.034	0.140	0.232
10	0.148	0.234	0.086	0.798	0.000	0.091	0.654	0.934	0.000	0.648	0.359	0.340

Table 4 Data from each run in the DoorKey environment (with the number of demonstrations provided to the agent). The highest mean success rate was observed in the stage that was provided with 10 demonstrations.

DoorKey environment for the 10 demonstration stage was statistically significant compared to that seen in the 4 demonstration stage, $p < 0.05$. The data is displayed in Table 4.

The 6x6 Dynamic Obstacles Environment produced a return of -0.996 when 4 demonstrations were provided (a result of -1 is recorded when the agent collides with the obstacle). The Dynamic Obstacles Environment produced a return of -0.127 when 6 demonstrations were provided. This improved slightly to a yield of -0.098 when increasing to 8 demonstrations. Further improvement to 0.051 was observed in the Dynamic Obstacles Environment when 10 demonstrations were provided. It should be noted that despite improved success with increasing the quantity of demonstrations provided in the Dynamic Obstacles Environment, the overall success numbers were still low. Still, statistically significant differences were observed in this environment between the 4 demonstration yields when compared to each of the other 3 stages of 6, 8, and 10 demonstrations, $p < 0.05$. The data is displayed below in Table 5.

Discussion

Our intention in this study was to assess the proficiency of T-REX IRL when the demonstrations provided were not directly suited to the algorithm in use[6]. While prior studies have described the efficacy of IRL learning in grid settings when T-REX is provided with consistently poor demonstrations, this is not easily extrapolated to demonstrations generated naturally[4,6]. To make this process more organic, we attempted to alter the quality and number of demonstrations to assess if these positive learning results could be replicated. We discovered that our results markedly differed based on the level of both the number of demonstrations provided and the complexity of the challenge placed on the agent. Our first environment, 6x6 Empty, was the least challenging for the agent. In the Empty environment, we showed that increasing the number of demonstrations provided improved the return yield. Notably, in both the 4 demonstration and the 6 demonstration stages, some runs returned nearly 0 scores. Those poor scoring runs in the 4 and 6 demonstration stages significantly decreased their respective mean success rates. Furthermore, the lack of errors in the 8 and 10 demonstration stages could have resulted from the additional input data that decreased agent error. We infer that more demonstrations provided in the Empty environment, as seen in the 8 and 10 demonstration stages, correlate with a lower chance of failure for any single run, and, therefore a higher mean success rate. As the stress level of the MiniGrid environment increased our data shows that the algorithm had greater difficulty attaining successful outcomes. In the 6x6 DoorKey environment very low levels of success were seen in the 4,6, and 8 demonstration stages. This was evidenced in that many agent runs would "give up" after producing an undesirable yield (score of zero), even after 50-100 steps. After observing these trials, the agent repeatedly had difficulty learning how to open the door that would lead it to the green ending zone. Improved yield was seen in the DoorKey Environment by increasing to 10 demonstrations, with only 2 runs failing to produce a score. Although the absolute degree of success was lower than seen in the less stressful Empty Environment, our results in the DoorKey environment also showed that yield improved with each incremental addition of demonstrations. The 6x6 Dynamic Obstacles environment differed from the prior two environments in that the algorithm could yield three scores - a positive score if successful, a zero score if the agent failed to reach the endpoint, or a negative score if the agent collided with an obstacle. Our data showed that the fewer demonstrations that were provided, the more likely an obstacle would collide with the agent. When 4 demonstrations were provided, the agent rarely reached the green ending zone without being hit by an object. Significantly

# demos	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Average Return	+ SD
4	-1.000	-1.000	-1.000	-1.000	-0.992	-1.000	-0.980	-1.000	-1.000	-0.986	-0.996	0.007
6	0.000	0.000	0.000	0.000	-0.032	-1.000	0.000	0.000	0.000	-0.238	-0.127	0.300
8	0.000	0.000	0.000	-0.010	-1.000	0.002	0.000	0.000	0.000	0.032	-0.098	0.301
10	0.000	0.148	0.009	0.000	0.000	0.000	0.043	0.108	0.000	0.204	0.051	0.071

Table 5 Data from each run in the Dynamic Obstacles environment (with the number of demonstrations provided to the agent). The highest mean success rate was observed in the stage that was provided with 10 demonstrations.

fewer collisions occurred in the 6 and 8 demonstration stages, yielding progressively less negative mean scores, but still very low levels of success. In the 10 demonstration stage the agent was able to avoid collisions more effectively, but could only produce low levels of positive successful mean outcomes. Still, in the Dynamic Obstacles Environment our data showed that increasing the number of demonstrations provided correlated with improved agent outcome.

Conclusions

These results supported our hypothesis that when provided with natural data, the success rate of our agents would directly correlate to the number of demonstrations provided. While this correlation remained present in all three MiniGrid environments, the mean yield success rates were lower as the environmental stress level increased. It can potentially be concluded from our results that further increasing the number of demonstrations provided in more stressful environments could enhance the efficiency of the IRL process in MiniGrid. Future research should evaluate adding even more demonstrations, while using greater computer power to assess if agent success rates and learning improve. Future work could also further attempt to quantify the impact of natural data demonstration use on IRL by comparing our findings to both a baseline Reinforcement Learning model and also to the original T-REX IRL implementation that utilized prioritized demonstrations.

List Of Abbreviations

IRL: Inverse Reinforcement Learning
T-REX: Trajectory-ranked Reward Extrapolation
AI: Artificial Intelligence

Declarations

Ethics, Approval, & Consent To Participate

Not Applicable.

Consent For Publication

Not Applicable.

Availability Of Data and Materials

The datasets used and analyzed during the current study are available from the corresponding author on reasonable request.

Competing Interests

The author declares that he has no competing interests.

Funding

Not Applicable.

Author's Contributions

ZP conducted the entirety of the research, analyzing and interpreting the data.

Acknowledgments

Thank you to the Farama Foundation for allowing us to use its public program MiniGrid, which provided a customizable, user-friendly platform to execute our trials. Discussions and insight provided by Tyler Moulton during the writing process were also greatly appreciated.

Author

Zac Penson is a Senior at North Broward Preparatory School in FL. He has an academic focus on computer science and participates in competitive coding contests. Outside school, he volunteers to teach coding to other high school students at the Family Impact of Lake Worth College Prep Program.

References

1 Santa Clara University, *SCU*, 2022.

2 D. Takeshi, *GitHub*, 2021.

3 J. Alexander, *The Gradient*, 2018.

4 A. Pablo, *Diva Portal*, 2019.

5 A. Kakko, *Alphanome.AI*, 2023.

6 D. Brown, W. Goo, P. Nagarajan and S. Niekum, *arXiv*, 2019.

7 A. Ng and S. Russell, *Stanford University*, 2000.

8 M. Chevalier-Boisvert, *GitHub*, 2024.

9 Francidellungo, *GitHub*.

10 M. Beliaev and R. Pedarsani, *arXiv*, 2024.