Review of "Learning Human Objectives by Evaluating Hypothetical Behavior" by Reddy et al.

Dr. Feng Luo, *Clemson University* Rajat Sethi, *Clemson University*

Abstract

In their article, "Learning Human Objectives by Evaluating Hypothetical Behavior," by Reddy et al., the research team discusses a new approach to reinforcement learning (RL). Unlike a standard RL algorithm, where a reward function is already known and implemented, this paper discusses a method that trains the reward model itself, then implements it into an agent. This idea has expanded and evolved into "ReQueST" (Reward Query Synthesis via Trajectory Optimization), a model that evaluates user feedback to properly create a reward system for an RL agent. Through this methodology, Reddy et al. has laid a foundation for applying RL to morally questionable scenarios without needing the agent to attempt them.

1. Description of Problems

Using ReQueST, the team sought out to solve three distinct problems.

1. How can we establish more informative labels than a standard policy while training?
2. How can we prevent "reward hacking" and false positives in reinforcement learning?
3. How can we prevent agents from falling into "unsafe" states?

1.1. Efficient Informative Labeling

In reinforcement learning, most settings have a definite reward function (i.e., Chess AI rewards winners, punishes losers, and does nothing for draws). However, some settings may not have such an obvious reward model. Many ethical games, such as the trolley problem and the MIT Moral Machine, are excellent examples of such ambiguity, especially when their scope expands to a real-world scale. This reward-based uncertainty becomes the main focal point of this research paper. How does a machine make rational decisions when the humans training them are unsure?

1.2. Reward Hacking Reduction

Another issue in RL is known as "reward hacking," in which an agent enters a state of unintended behavior because the reward system faultily gives points for the wrong actions. Moreover, this problem is exacerbated for morally ambiguous scenarios since the reward model would change depending on what ethics the developer values most. In this case, the team must make sure that the "false positive" rate is as low as possible, matching what the developer wants with what they get.

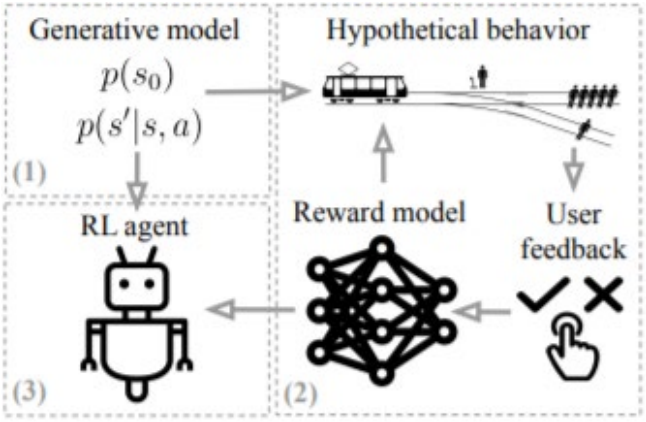
1.3. Unsafe State Avoidance

One final aspect that the team considered was the necessity to avoid "unsafe" states while testing the RL agent. In most RL settings, the agent is allowed to enter an unsafe state and fix its mistakes. However, this only applies when the stakes are low, like during a Chess game. In a real-world, risky scenario, the agent does not have any leeway to make a mistake, do something dangerous, and rectify its mistake. For example, if the algorithm is driving an autonomous car, hitting a pedestrian would be an "unsafe" state. Once the car enters this state, the damage has already been done, posing disastrous consequences. To prevent such an outcome, the algorithm must predict that its about to enter an unsafe state and swerve its decision-making to a better one.

2. Review and Analysis of Solutions

2.1. Reward Model - Review

The first major component of ReQueST is its reward model, best illustrated by the following diagram found in the paper (Figure 1).



Section 2 of the figure illustrates the study's main contribution. Instead of a static reward function, ReQueST uses a dynamic reward model fueled by user feedback. The user is given a hypothetical scenario, then labels each decision with a reward. That information is subsequently inputted into a neural network which will eventually become the RL agent's reward function. This process of collecting user data

repeats until the RL agent asymptotes in false positives and perfectly avoids every unsafe state before they happen.

2.2. Reward Model – Analysis

While the idea of training a reward model is not novel, it is still experimental and shows promise. In each environment, ReQueST has been able to outperform random policies and simple reward-maximizing trajectories. However, ReQueST still underperforms when compared to an offline, predetermined reward model, so there is still plenty of room for improvement.

2.3. Acquisition Functions – Summary

When determining what query trajectories the user will analyze, the research team prepared four acquisition functions (AF) to prioritize certain goals over others.

- Maximize Trajectory Uncertainty (Gives the user tougher queries)
- Maximize Trajectory Novelty (Gives the user more diverse queries)
- Maximize Rewards
- Minimize Rewards

Further, these acquisition functions can be combined to create "Hybrid AFs," such as combining "Trajectory Uncertainty" with "Reward Maximization," depending on what the model needs the most.

2.4. Acquisition Functions – Analysis

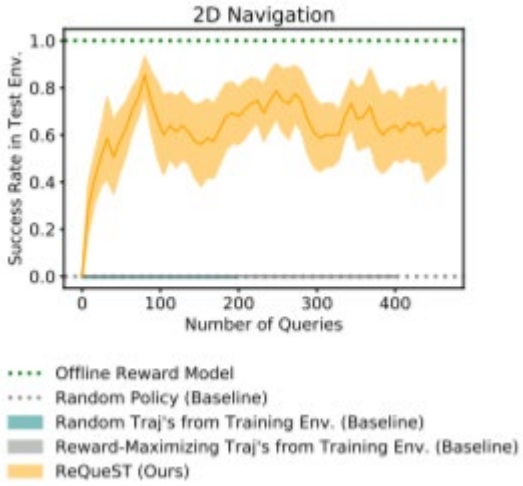
To my knowledge, these types of AFs are quite standard in Bayesian Optimizations (Frazier). The research team has already performed an ablation study to determine the importance of each individual AF in each environment (Section 4.6). In future analysis (likely Phase 3), I will conduct a more robust scrutiny of the AFs while simultaneously generating some of my own to improve the current model.

3. Personal Insights and Thoughts

While reading the article, two aspects stood out as potentially misleading, their comparison baselines and the types of training environment used. For future reference, these two issues are where I will most likely focus on for the other phases of this project.

3.1. Questionable Baselines

In the "Experimental Evaluation" section of the paper, the team compared ReQueST's performance to a few control groups or baselines. The main issue is that some of these comparisons were meaningless and could not accurately reflect ReQueST's abilities. Take for example the following graph, which was supposed to show how the model did in a simple 2D navigation environment (Figure 4).



As the diagram illustrates, there are four control groups, the Offline Reward, the Random Policy, Random Trajectories, and Reward-Maximizing Trajectories. The problem is that none of them provide an adequate comparison. The Offline Reward is already well-established for simple navigation with a 100% success rate, and the other three control groups are about as useful as doing nothing, since they have 0% success rates. This is a somewhat recurring issue in the paper, so when I write my own final report, I will attempt to remedy this result comparison problem.

3.2. Limited Application Environments

The other issue I had with this paper involved the types of training environments used. The research team used their model in three situations, MNIST digit recognition, 2D navigation, and Car Racing. While these environments are classical examples for RL, they were not applied to advanced ethical dilemmas. In other words, these examples did not adequately show how ReQueST can be utilized for risky situations with ambiguous reward functions, contrary to what the abstract implied. In future phases, I intend to apply ReQueST to an environment where its reward model can be stress-tested to a greater degree.

4. References

Frazier, P.. "A Tutorial on Bayesian Optimization." *ArXiv* abs/1807.02811 (2018): n. pag.

Reddy, Siddharth & Dragan, Anca & Levine, Sergey & Legg, Shane & Leike, Jan. (2019). Learning Human Objectives by Evaluating Hypothetical Behavior.