Path planning problem on virtual map based on RLHF

Yi Lu 121090386, Yidan Wang 223040053, Fengyuan Ye 120090367, Jinyu Xie 121040075

Abstract

RLHF (Reinforcement Learning with Human Feedback) involves collaborative learning between machine agents and humans. By incorporating human feedback, the agents refine their strategies and enhance overall performance. This approach finds applications in diverse areas, including autonomous decision-making and path planning, fostering effective human-machine collaboration. In this paper, we observe the path maps generated by the model, providing feedback to enhance its alignment with human preferences. Instead of solely pursuing reward maximization, we emphasize the importance of incorporating user needs. Additionally, we introduce the concept of providing the model with optimal paths for learning, allowing it to derive results that better align with user requirements.

1. Background

This chapter is divided into two parts. The first part introduces how to plan paths on a virtual map. The second part introduces the basic theory and application scenarios of the RLHF algorithm. Finally, it introduces the reason to use RLHF for path planning tasks.

1.1. Path planning

Path planning involves finding the most efficient route from a starting point to a goal, crucial in optimizing navigation through diverse environments. By choosing a sequence of steps in space, path planning enables systems to effectively navigate, avoiding obstacles and adhering to specific constraints or costs.

Its applications span across various domains, including autonomous driving, robot navigation, and logistics management, where it plays a pivotal role in delivering effective and secure navigation solutions. Whether guiding autonomous vehicles, facilitating robot movement, or optimizing logistics routes, path planning remains integral in enhancing efficiency, safety, and overall system performance.

1.2. Introduction of RLHF

Reinforcement learning[1] is a machine learning paradigm. In this approach, an agent learns an adaptive policy by executing actions in the environment. The agent receives reward signals to maximize cumulative reward. RLHF differs from traditional reinforcement learning. It involves incorporating intuitive feedback from human experts or users into the learning process. This involvement of human feedback and learning allows the system to adapt more flexibly to actual needs. It results in improved performance and promotes more effective collaboration between humans and machines.

RLHF has broad development prospects and potential in the field of artificial intelligence. Its development prospects can be summarized as follows:

(1) Intelligent decision-making and control[2]: RLHF enables machine agents to learn from human feedback, which has important applications in fields such as autonomous decision-making, path planning, autonomous driving, and intelligent robots.

(2) Human-machine collaboration[3]: RLHF encourages that human cooperate with machines, which provides real-time feedback to help machines complete tasks better.

(3) Automation field: In the field of automation, RLHF can improve the performance of autonomous robots and robotic systems, including drones, robotic manufacturing and logistics management.

In summary, RLHF represents the frontier of reinforcement learning and human collaboration and has broad potential to accelerate the development of artificial intelligence. In the future, we can look forward to more innovations and applications to fully realize the potential of RLHF in different fields.

2. Related Works

This section mainly introduces the existing path planning algorithms and the practical applications of RLHF.

Early path planning algorithms are basically based on search algorithms, such as the A*[4] algorithm proposed by Peter E. Hart et al., which is a heuristic search algorithm. It is used to find the shortest path from the starting point to

the target node in the graph and guide the search process by utilizing the estimated heuristic information. In the A* algorithm, it uses a heuristic function to estimate the cost from the current node to the target node, and selects the path based on the actual cost that has been traveled. The A* algorithm is widely used in path planning, game development and other fields. And the Dijkstra algorithm [5] proposed by EW Dijkstra is also widely used in path planning tasks. However, these algorithms exhibit certain limitations. They are not well-suited for high-dimensional state spaces, and dynamic environments present challenges for their effective application.

Researchers have improved path planning in high-dimensional and complex environments and proposed RRT (Rapidly-Exploring Random Trees)[6], a probabilistic path planning method for dealing with dynamic environments and high-dimensional state spaces. PRM (Probabilistic Roadmap)[7] builds a graph through random sample points and is suitable for high-dimensional spaces and complex environments. The adaptability to high-dimensional state spaces is limited. The processing capabilities for dynamic environments are relatively weak. When building a graph structure, space needs to be sampled, which may lead to sampling bias and incompleteness of the graph.

When reinforcement learning is more and more popular, reinforcement learning methods also have been successful in virtual map path planning. However, they are usually trained in virtual environments and have difficulty adapting to the complexity of the real world. Researchers proposes human-machine collaborative path planning: RLHF allows human and machine agents to collaborate, provide real-time feedback, and improve path planning performance. This collaborative approach shows great potential in improving path planning performance and user experience. RLHF is also used in simulation environment and game design. The RLHF method is used to improve the path planning capabilities of characters in the game and provide a more realistic and intelligent virtual experience.

This project aims to integrate RLHF and virtual map path planning to provide a smarter and more adaptable path planning solution. Continuous innovation in this field provides new possibilities for intelligent navigation of autonomous robots and virtual agents.

3. Method

This section mainly introduces our algorithm flow, which includes environment setting and implementation process.

3.1. Map Setting

The main points of the virtual map path planning problem include:

1. Start and end points: The input to the problem includes the start position and the target position. The goal of the path planning algorithm is to find the best path which will maximize reward connecting these two locations. In this maze, the starting point is (0, 0) and the end point is (5, 9). As shown in 1 , the yellow point is the starting point and the black point is the end point.

2. Map representation: The intricately designed map featured in this article spans across a 10*10 grid, offering a detailed representation of a dynamic environment. It goes beyond a mere depiction of spatial layout, incorporating essential elements like obstacles and rewards. This comprehensive map serves as a foundation for diverse scenarios and simulations, providing a rich and immersive backdrop for various applications.

3. Reward setting: We use gaussian noise to simulate the reward, so that each point in the path has a positive or negative reward. The mean of this Gaussian distribution is 0 and the standard deviation is 30. To ensure the repeatability of the experiment, we use np.random.seed(0) function in python to produce the same map in each simulation. As shown in 1 red represents negative rewards, which are obstacles, and blue represents positive rewards. Besides, the darker the red or blue color is, the greater the value of positive or negative reward is. And reaching the destination will provide a reward of 10,000, which is much larger than any other node's reward. Consequently, our goal is train the agent to reach the destination.

Our algorithm first creates a map of size 10x10 with randomly generated rewards and obstacles on the map. Each tile in the map contains a generated value that represents the strength of a reward or obstacle. Next, by mapping these values to colors. The example is as follows.

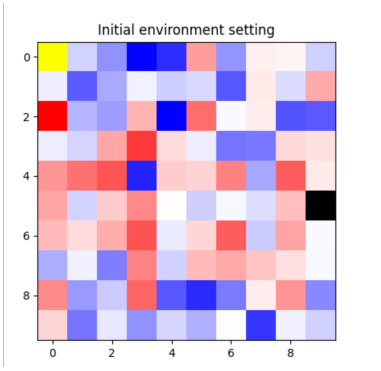

Figure 1. A picture of the initial map

3.2. Q-learning

Q-learning is an algorithm based on reinforcement learning, first proposed by Christopher J. C. H Watkins in 1989. The algorithm is designed to allow the agent to learn optimal strategies through interaction with the environment to maximize accumulated rewards. In virtual map path plan-

ning, Q-learning shows significant advantages. Its power lies in its ability to learn autonomously in an unknown environment without the need to accurately understand the environment model. It makes it more flexible in practical applications and suitable for complex and dynamic scenarios.

Path planning is a key task of Q-learning in virtual map applications. By exploring states and updating strategies on the map, Q-learning enables the agent to gradually learn and optimize path selection to achieve the established goal. Its characteristic of requiring no prior knowledge makes it ideal for handling path planning in unknown or dynamic environments. The motivation for using Q-learning for path planning is that through continuous trial and error, it can gradually improve the decision-making ability of the agent, allowing it to deal with various complex situations more effectively, providing an adaptive method for path planning and efficient solution. The algorithm pseudo-code is as follows 2:

Algorithm : Q-learning

Input: ϵ, α

For all $(s,a) \in \mathcal{S} \times \mathcal{A}$, $Q'(s,a) \leftarrow 0$, $Q(s,a) \leftarrow \infty$

while $\|Q - Q'\|_\infty > \epsilon$ **do**

  $Q \leftarrow Q'$

  Sample a trajectory τ from the policy $\pi(a \mid s) = \arg\max_{a \in A} Q(s,a)$

  For all state-action-reward-state tuple $(s,a,r,s') \in \tau$,

  $Q'(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \max_{a' \in \mathcal{A}} [r + \gamma Q(s',a')]$

$Q^* \leftarrow Q$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$

$\pi^* \leftarrow \arg\max_{a \in \mathcal{A}} Q(s,a)$

return $Q^*(s,a)$, $\pi^*(s)$ for all s,a

Figure 2. The pseudo-code of Q-learning

We begin by setting the Q-table values to zero. As the process unfolds, the Q-table experiences ongoing updates after exploration. The learning rate, represented as alpha, functions as a parameter measuring the magnitude of these updates. A higher alpha signifies a more significant adjustment of Q-values. Besides, epsilon measures the agent's openness to try out random actions, even when a potentially optimal action is present in the Q-table. A heightened epsilon indicates a greater tendency of the agent to explore the environment.

3.3. Add Human Feedback

When simply using Q-learning, the agent can easily fall into a situation where it only pursues high returns without reaching the destination point. For example, if the rewards of two neighbor points are very high, then the agent will wander between these two points instead of choosing to go to the destination.

There are two reasons of being stuck between two nodes with high rewards, so-called comfort zone of the map, without progressing for the destination for the agent. Firstly, it can be really desirable for the agent to do so at least locally since it can't figure out a better path in the local environment. In addition, the nodes around the destination node has lower rewards, which implicitly prevent the agent to approach the destination node.

What we are doing here is deliberately pushing the agent beyond its accustomed boundaries, a strategy laden with risks. This approach may yield one of three potential outcomes: the agent might seek refuge in a different comfort zone, successfully navigate towards the intended destination, or revert back to its initial state of comfort zone. This intentional disruption of the agent's comfort zone introduces an element of unpredictability, opening up a spectrum of possibilities that could shape the agent's behavior in diverse and intriguing ways.

After adding human feedback, it can probably help the agent get out of the current predicament. Although the probability of successful "rescue" is relatively low when human intervention is small, it is already much better than the training result without human feedback. Additionally, whether the agent can reach the end depends largely on the input of human feedback.

By combining the traditional machine learning method Q-learning with human feedback and applying it to the path planning problem, significant improvements were achieved. This innovative method realizes the collaborative cooperation of machine and human intelligence, achieving superior performance in the field of path planning. This hybrid method leverages the reinforcement learning capabilities of Q-learning while making full use of the intuitiveness and professionalism of human feedback, thereby achieving satisfactory results in improving the accuracy and efficiency of path planning. This human-machine cooperation strategy not only enriches path planning solutions, but also improves system performance and enhances user experience.

4. Results display and analysis

When the agent gets stucked between high reward nodes, we can add human feedback. In this experiment, we set up this prompt statement to guide the user: Please enter one action (0: go up one step, 1: go down one step, 2: go left one step, 3: go right one step) Enter quit if the human feedback is done.

As can be seen from the table below1, models that add human feedback will receive higher rewards because it reaches the destination and receives the largest reward so that it gets a higher reward in total.

Method	Average Reward
Q-learning (Iteration 1000 times)	630.67
Adding Human Feedback	3877.14

Table 1. Results. Ours is better.

The following six pictures3 all show that the agent is trapped between (0, 4) and (0, 5), and has not reached the destination. This is because the rewards at these two points are relatively high, and the agent will fall into the local optimum, and give up looking for the destination. This is not a desirable result, definitely.

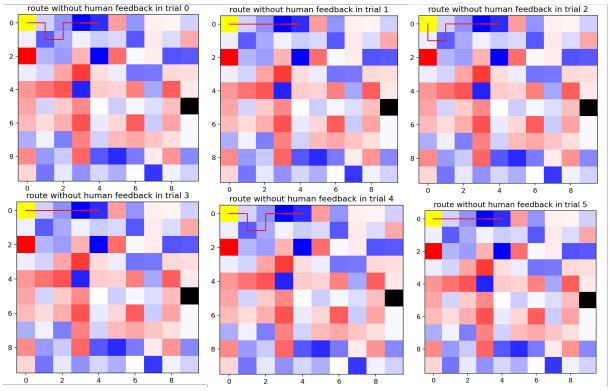

Figure 3. The result without human feedback

As shown in the figure4, the red route represents the path taken by the model through Q-learning, and the green route represents the path that the agent is forced to go under human guidance. It can be seen that after adding human feedback, the agent will be likely to get out of trouble. Although it is not successful every time, it is already a big breakthrough.

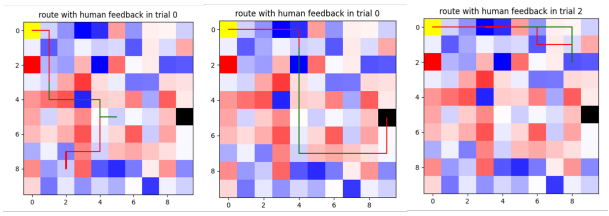

Figure 4. The result with human feedback

5. Conclusion

In our project, we meticulously assess the efficacy of the algorithm-generated path maps, offering insightful recommendations to enhance their alignment with individual preferences. Departing from the exclusive pursuit of reward maximization, we emphasize the paramount importance of seamlessly integrating user-specific requirements into the algorithmic framework. Furthermore, we introduce the innovative concept of furnishing the algorithm with optimal paths for the learning process, thereby empowering it to generate outcomes that align more closely with user needs.

Q-Learning, being a model-free approach, operates without the necessity of a predefined environmental model, rendering it adaptable to diverse environments. Notably, this algorithm dynamically refines its strategy based on real-time feedback from the environment, facilitating efficient learning in various settings.

Even though we adopt 10*10 map in our experiment, the size of the map can be easily extended into m*n for arbitrary integer m,n. Namely, it is applicable for any rectangle size of map.

However, it is imperative to acknowledge certain limitations in our research. The extensive interactions and time investment required for effective learning may prove impractical in complex environments. Additionally, the escalating complexity of path planning challenges can result in a substantial growth in both state and action spaces. Last but not least, we hope to find a human guided path as short as possible because we hope to input as few human guidance as possible to utilize the power of machine learning. Nevertheless, it is difficult to provide a reasonable and short human guided path efficiently for arbitrary dilemma node in the map.

To address these limitations in the future, the implementation of more sophisticated strategies holds the potential to substantially enhance the efficiency and effectiveness of the learning process. This forward-looking perspective underscores our commitment to refining and advancing the capabilities of path planning algorithms in order to meet the evolving demands of diverse and intricate environments. We currently only use one method as the human feedback approach. In the future, we can give a score for the paths generated by the model and then feed it back to the model to obtain to train a better model. However, this method is more difficult to implement, so it can be practiced in subsequent experiments.

References

[1] Thrun S, Littman M L. Reinforcement learning: an introduction[J]. AI Magazine, 2000, 21(1): 103-103.

[2] Hadfield-Menell D, Russell S J, Abbeel P, et al. Cooperative inverse reinforcement learning[J]. Advances in neural information processing systems, 2016, 29.

[3] Karur K, Sharma N, Dharmatti C, et al. A survey of path planning algorithms for mobile robots[J]. Vehicles, 2021, 3(3): 448-468.

[4] Hart P E, Nilsson N J, Raphael B. A formal basis for the heuristic determination of minimum cost paths[J]. IEEE transactions on Systems Science and Cybernetics, 1968, 4(2): 100-107.

[5] Dijkstra E W. A note on two problems in connexion with graphs[M]//Edsger Wybe Dijkstra: His Life, Work, and Legacy. 2022: 287-290.

[6] LaValle S M, Kuffner Jr J J. Randomized kinodynamic planning[J]. The international journal of robotics research, 2001, 20(5): 378-400.

[7] Kavraki L E, Svestka P, Latombe J C, et al. Probabilistic roadmaps for path planning in high-dimensional configuration spaces[J]. IEEE transactions on Robotics and Automation, 1996, 12(4): 566-580.