Performance Comparison of Function Approximation-based Q Learning Algorithms for Autonomous UAV Navigation

Huy Xuan Pham, Hung Manh La, David Feil-Seifer, Luan Van Nguyen

Abstract—Using reinforcement learning to enable Unmanned aerial vehicles (UAV) carry out missions in unknown environments in which their mathematical model may not be available, is an active research topic. However, there is a challenge in implementing RL for real-world applications. This paper provides a framework for using RL to allow the UAV to navigate successfully in such environments. A performance comparison of three different Q learning methods: classical Q Learning (QL), Fixed Sparse Representation-based Q Learning (FSR-QL), and Radial Basis Function-based Q Learning (RBF-QL), is presented. We conducted simulations to show how the UAVs can successfully learn to navigate through an unknown environment. Through the simulation comparison of these three different learning methods (QL, FSR-QL, RBF-QL), the RBF-QL outperforms the others in term of space reduction and learning speed (convergence time).

I. INTRODUCTION

Unmanned aerial vehicles (UAV), or drones, in missions involving navigating through unknown environments is becoming more popular in many applications, as they can host a wide range of sensors to measure the environment with relative low operation costs and high flexibility. Most applications normally make assumptions on the model describing the target, or prior knowledge of the environment [1], despite the fact that the knowledge and data regarding the environment are usually not accurate or available. Learning algorithms, such as reinforcement learning (RL), offer an alternative approach to conventional control methods, as they allow the UAV to learn the right behaviors through the interactions with the environment [2].

Applying RL algorithms to UAV applications has long been studied. They have been demonstrated effectively in UAV control problem, for example, achieving desired trajectory tracking/following [3], path planning [4], or carrying loads [5]. But most of the existing work often neglects important issues in implementing the algorithm in real application. One particular issue is representing the state - action value function in continuous space [6].

This paper aims to highlight the impact of approximation in RL algorithms, through a UAV navigation problem. We first implemented a classical Q-learning algorithm to help the drone navigate successfully to a goal without knowing its exact position. Then, we discussed how it can be improved with function approximation techniques, Fixed Sparse Representation (FSR), and Radial Basis Function (RBF), to deal with large Q table size. The remainder of this paper is organized as follows. Section II provides details on the problem statement, and the approach we use to solve the problem. Basics in RL algorithm are discussed in section III. We address the importance of function approximation techniques in section IV. A simulation is conducted on section V to study the impact on performance of these algorithms, and finally, we draw conclusions in section VI.

II. PROBLEM STATEMENT

Consider a simple problem of navigation, where a UAV, such as a quadrotors, needs to find a target (Figure 1) in an unknown environment. The exact position of the goal is unknown to the UAV, but it can be recognized based on a special landmark.

Fig. 1. A UAV navigation in unknown environment. Blue circles represent discretized state space, while the goal is marked by a red circle. The exact position of the goal is unknown to the UAV, but it can be recognized based on a special landmark.

In RL, an agent learns a good behavior through interactions with the environment, judging its action given at a state by observing a reward signal from its surrounding, with a purpose to find an optimal policy that allow it to receive maximal reward over the learning course [2]. We defined a learning model assuming Markov Decision Process property as a tuple \( < S, A, T, R > \), where \( S \) is a finite state list, and \( A \) is a finite set of actions. \( s_k \in S \) and \( a_k \in A \) are the state
and the action that the agent takes at step $k$, respectively; $T$ is the transition probability function, $T : S \times A \times S \to [0, 1]$, describing the probability of the agent that takes action $a_k$ to move from state $s_k$ to state $s_{k+1}$. In a deterministic setting, $T(s_k, a_k, s_{k+1}) = 1$. $R$ is the reward function: $R : S \times A \to \mathbb{R}$ that quantifies the immediate reward of the agent for getting to state $s_{k+1}$ from $s_k$ after taking action $a_k$. We have: $R(s_k, a_k) = r_{k+1}$.

The agent in the system uses a state - action value function $Q(s_k, a_k)$ to measure the performance of an action in a given state, so it can determine the right action to take. A popular RL algorithm, known as classical Q-learning (QL) [8], allows the agent to iteratively compute the optimal value function. In this work, we employ two different approximation techniques called Fixed Sparse Representation (FSR), and Radial Basis Function (RBF) to map the original Q table to a parameter vector $\theta$. In FSR-QL, each element of function $\phi$ is defined as:

$$
\phi_l(s_k, a_k) = \left\{ \begin{array}{ll}
1, & \text{if } s_k = S_l \in S, a_k = A_j \in A \\
0, & \text{otherwise}
\end{array} \right.
$$

(3)

For RBF-QL, each element of function $\phi$ is defined as:

$$
\phi_l(s_k, a_k) = \left\{ \begin{array}{ll}
\frac{1}{\sqrt{2\pi\sigma_l^2}}, & \text{if } a_k = A_j \in A \\
0, & \text{otherwise}
\end{array} \right.
$$

(4)

where $c_l$ and $\mu_j$ are the center and radius of the RBF $l$, which has the shape of a Gaussian bell. Pseudo codes for FSR-QL and RBF-QL are presented in Algorithm 1, modified from [6].

We can calculate the space saved with FSR and RBF techniques for a problem with state space $S = D_x \cdot D_y$, where $D_x$, $D_y$ are the dimension of $x$, $y$ coordinates, respectively. We use $|\cdot|$ to denote the number of element of a vector. If we use FSR as in equation (3), both $\phi(s, a)$ and $\theta$ are column vectors of the size of $(D_x \cdot D_y) \cdot |A|$, which is much less than the space required in the original Q-table, which requires the size of $|S| \cdot |A| = (D_x \cdot D_y) \cdot |A|$. For instance, if the environment space is $S = 10 \times 10$, the original Q-table needs space to store $10^2 \cdot 4 = 400$ distinct values, while the total space to store the approximated parameter vector $\theta$ requires only $(10 + 10) \cdot 4 = 80$ distinct values. The saving ratio, that is $\frac{400}{80}$, will grow as the state space grows. In RBF, we can potentially save even more space, since we can use a more limited, fixed number of basis functions to represent the state space. Figure 2 shows such a configuration where we use $n = 4$ RBF, with centers at $[2,2]$; $[2,4]$; $[4,2]$; $[4,4]$, and radius $\mu = 2$. The total space to store the approximated parameter vector $\theta$ requires only $n \cdot |A| = 4 \cdot 4 = 16$, much less than the parameter size in FSR-QL and original Q table size in QL.

### V. Simulation

This section provides details of simulations on MATLAB for the UAV navigation problem. We defined the environment as a 5 by 5 board (Figure 2). Suppose that the altitude of the UAV was constant, the environment is discretized into 25 states, from (1,1) to (5,5). The UAV needs to find a goal, located at (5,5) but unknown to the UAV, from starting...
Each simulation used identical learning parameters: learning rate $\alpha$, discount rate $\gamma$, and radius $\mu$. The first RBF-QL used 4 RBF with centers at $[2, 2]; [4, 2]; [4, 4]$, and radius $\mu = 2$ (Figure 2). The second RBF-QL used 9 RBF with centers at $[2, 2]; [2, 4]; [4, 2]; [4, 4]; [1, 3]; [3, 3]; [5, 3]; [3, 1]; [3, 5]$. Each simulation used identical learning parameters: learning rate $\alpha = 0.1$, and discount rate $\gamma = 0.9$.

Figure 3 shows the resulting performance of the four learning instances. The black dotted line shows the number of steps in each episode for RBF-QL with 4 RBF. Although it took more episodes (18) to converge than FSR-QL, it is still much less than the classical QL. The red solid line, that shows the number of steps in each episode for RBF-QL with 9 RBF, converged after 10 episodes, which is close to FSR-QL speed performance, while saving more space than the latter method. It can be noted that the RBF-QL performs better in convergence speed if more basis functions are used, since it would increase the accuracy of the approximation of the value function.

VI. CONCLUSION

This paper presented a performance comparison of different Q learning algorithms: QL, FSR-QL, and RBF-QL with different configurations that can help a UAV navigate to the target point in an unknown environment. The simulation showed that the Q learning with function approximation algorithms (FSR-QL, RBF-QL) outperforms the original QL in term of learning speed (convergence time). The RBF-QL offered the most significant space reduction. The result is not limited to the simple UAV navigation problem presented in this paper, and can be used in implementing RL algorithm for more important applications, such as wildfire monitoring, or search and rescue missions.

REFERENCES