Report on Maze Rider

Hongyu Zhou Souhaiel Ben Salem Shiyao Li

1. Introduction

The Maze Rider problem is an example of a more generalized issue, the trade-off between exploration and exploitation. In this project, the existence of the wall and the pits, and the changing position in every episode make the problem more challenging. To tackle the exploration-exploitation problem, we devote ourselves to efficient sampling, reward shaping, state space reduction as well as exploring different models.

For this project, we have implemented several different algorithms for deep reinforcement learning on the Maze Rider problem. During the testing phase, we test the performance of not only the small map but also do relevant experiments and tests on the large map.

In the first exercise, we store the states (the position of the player and the position of the goal) in a hash in a compact way. Then, we compare the differences between Q-learning and its variants for a model-free tabular reinforcement learning algorithm. In the second exercise, we use DQN and policy-based methods separately. In the end, we can achieve good results in both the small map and the large map based on model-free algorithms. As for deep reinforcement learning, we have quick converging performance in the small map but sub-optimal performance in the large map.

2. Algorithms

2.1. Model free tabular RL

Before getting into tabular RL models, we construct a hash function to map the state into a more compact space. Considering that it is generally the agent position, the goal position and the pit positions that characterize the state, we tend to ignore the information for empty cells and walls. Since in each environment, the pits are selected following a Bernoulli distribution, the number of states increases exponentially with regard to the total number of pits, which may lead to the explosion of the state space. To this concern, it is a reasonable sacrifice that we select merely goal position and agent position as the state space, making the dimension linearly to the maze size and the total number of goal positions.

We implement Q-learning, double Q-learning, SARSA and Dyna-Q in this setting, with various sampling methods and reward shaping techniques as complementary ingredients.

2.2. Deep RL

We apply DQN, reinforce and A2C methods to tackle the problem. In DQN, we also implement various sampling methods and heuristic approaches such as reward reshaping to help explore the environment. In addition, we try different ways to formulate the state space such as MLP and convolution network. As for policy based method, same techniques are utilized but without a converging performance even in the small environment.

2.3. Sampling

In our tabular algorithms as well as in DQNs, the sampling method typically used is an epsilon-greedy exploration which we implemented first, it involves selecting the best action based on the current estimated Q-values with probability 1-ϵ, and selecting a random action with probability ϵ.

Considering the disadvantage of these algorithms is the lack of active exploration, we also try to use Thompson Sampling. In the Thompson Sampling algorithm, we need to maintain a prior distribution $p(\theta)$ and a posterior distribution to estimate the value of each action. In the Maze Rider game, we can use a neural network to estimate the value function for each action. After every time we select an action and observe a reward, we update the posterior distribution $p(\theta|D_{1:t})$, $D_{1:t}$ denotes the empirical data for the previous t time steps. And use the new posterior distribution to select the next action.

2.4. Reward shaping

We apply two methods to reshape the reward. Firstly we penalize the agent for hitting the walls. If the state and the next state are identical, then the agent hits the wall and we give it an additional negative reward. Secondly, we apply a count-based penalization at every step. In an episode, the more (state, action) tuple is visited, the more penalization we give to the agent, which is similar to giving the agent a short period of memory. These methods efficiently help the agent avoid hitting the wall and sticking itself in a loop.

3. Experiments

3.1. Effect of reward shaping

We do an experiment(with less episodes) in Q-learning and deep Q-learning to compare the performance with or without reward shaping. It is found that reward shaping can drastically accelerate the learning speed of the agent and its converging performance3.

3.2. Study on hyperparameters

The learning rate α1 and exploration rate ϵ2 are the two most important hyperparameters to study in tabular RL, where the state and action spaces are small and discrete. This is because the learning rate determines the step size of the Q-value updates, which affects the rate of convergence and the stability of the algorithm while the exploration rate affects the balance between exploration and exploitation.

3.3. Thompson sampling

There is no obvious benefits for Thompson sampling over ϵ-greedy sampling5.

3.4. Success in larger environment

The Q-learning learns slowly but converges successfully however the size of the environment. The Deep Q-learning learns faster, but in larger and more complex environment, it reaches sub-optimal points but not guaranteed to reach the optimal q-function4.

4. Discussion

SARSA tends to avoid dangers, which accounts for their poor performance in this specific environment where pits and goals can gather together. Q-learning or DQN, however, succeed in accomplishing the task because when they update the q-value, they take the maximum q-value of the next state, hence not influenced by negative actions at the next state. Reinforce algorithm fails because it is essential for it to have a complete episode, or the cumulative return is always 0. In addition, the changing position in every episode poses another challenge for learning in this algorithm. A2C fails for the same reason as in SARSA. When pits and goals stay close to each other, A2C tends to avoid danger.

In case of sparse reward tasks, it is useful to reshape the reward by making it dense. Counting based reward and penalization can better regularize the agent's performance. The agent also benefits from a contracted state space. Compared to tabular RL, deep RL has the advantage in the state space design. It is more flexible and more practical to contract the state space in deep RL. A proper learning rate and exploration ratio(epsilon) can accelerate the convergence speed, yet the final performance varies little.

5. Appendix

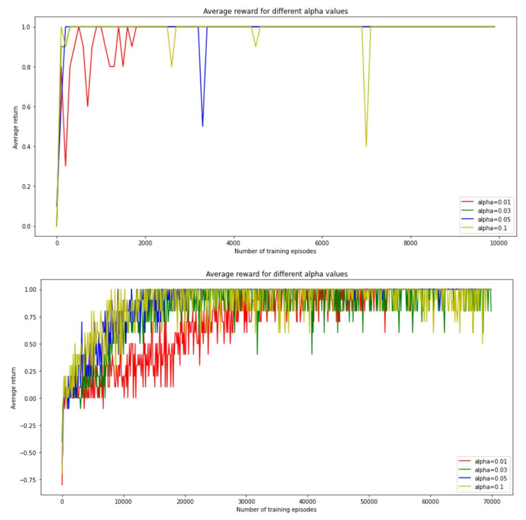

Figure 1. Comparison of the effect of α. Top: small env, Down: large env

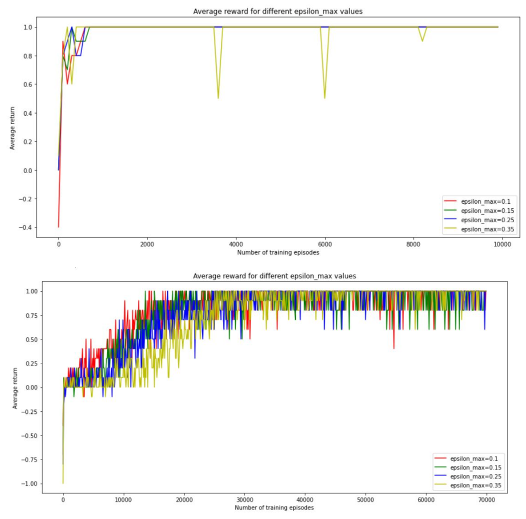

Figure 2. Comparison of the effect of ϵ. Top: small env, Down: large env

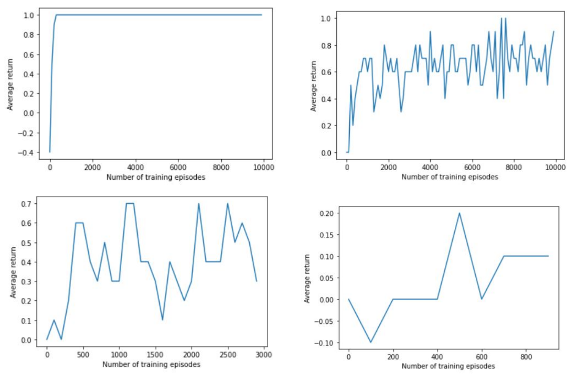

Figure 3. Comparison of the effect of reward shaping. Top left: small env QL with reward shaping, Top right: small env QL without reward shaping, Down left: large env DQL with reward shaping, Down right: large env DQL without reward shaping

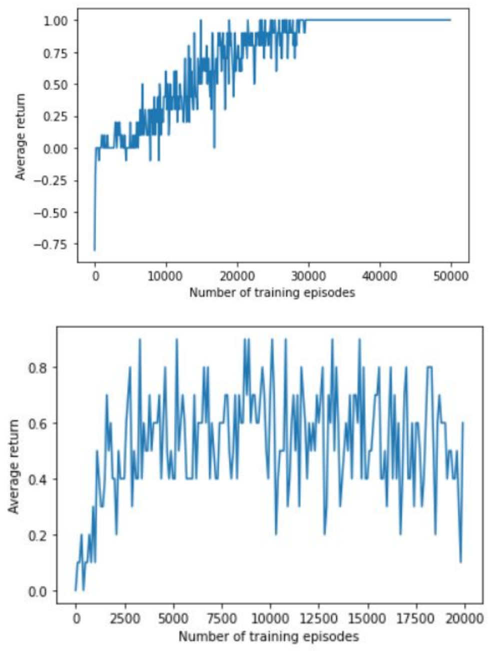

Figure 4. Success in large environment. Top: Q-learning, Down: Deep Q-learning

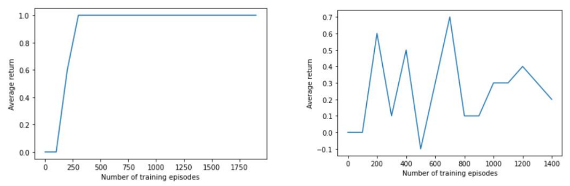

Figure 5. Thompson sampling in DQL. Left: small env, Right: large env