Deep RL Software project

Amine Oueslati

Eötvös Loránd University - AI specialization

Abstract. This report is for explaining the process of implementing reinforcement learning solution for the Frozen-Lake-v0 environment.

1 Introduction

In this task the "OpenAI Gym" environment used is the "FrozenLake-v0 ", and the agent will be learning through the SARSA algorithm.

2 Related work

"OpenAI Gym" is a toolkit that allows testing of different reinforcement learning algorithms on various simulated environments with the overall goal of maximizing the reward from interacting with that environment.

There are different available environments, from the simple grid-world type, rendered on the command line, to ones with more complex dynamics, and then the more difficult environments with many compound variables.

3 Environment

The agent controls the movement of a character in a grid world. Some tiles of the grid are walkable, and others lead to the agent falling into the water. Additionally, the movement direction of the agent is uncertain and only partially depends on the chosen direction in other words the ice is slippery, so the agent won't always move in the direction it intend.

The agent is rewarded for finding a walkable path to a goal tile. The episode ends when it reaches the goal or fall in a hole. it receives a reward of 1 if it reaches the goal, and zero otherwise.

4 Methods

At first, the Q-table is randomly initialized, and as the agent interact with the environment the Q values will be learned through small updates. Thus, one can think of the values derived from the Q function as the targets that the approximated Q values will be pushed towards at some learning rate.

However, the agent is selecting the actions greedily based on the randomly initialized Q-table and not exploring the other actions. To fix this issue, a new variable epsilon is introduced to balance the probability of exploration and exploitation of the environment. Nevertheless, this variable should be reduced as the agent learns a more optimal Q-table.

This algorithm could be implemented as a neural network too. And since the states and actions are discrete, it makes more sense to represent them as one-hot vectors. thus, the state dimensions will be expanded to have nS (number of states) columns and actions will be encoded to nA (number of actions) columns.

The output Q-values for an input state vector with nS columns, should have nA columns. Therefore, a dense layer is applied to achieve that. Afterwards, it is multiplied, element wise, by the one-hot action input to "zero-out" all the Q-values, except for the given one. Subsequently, the result is summed along the columns to get a single Q-value. the mean squared error is used for the loss function where the target is the result of the Q-function using the reward added to the maximum of the next state's Q-values, which is derived from the model.

This diagram can illustrate the topology of the network:

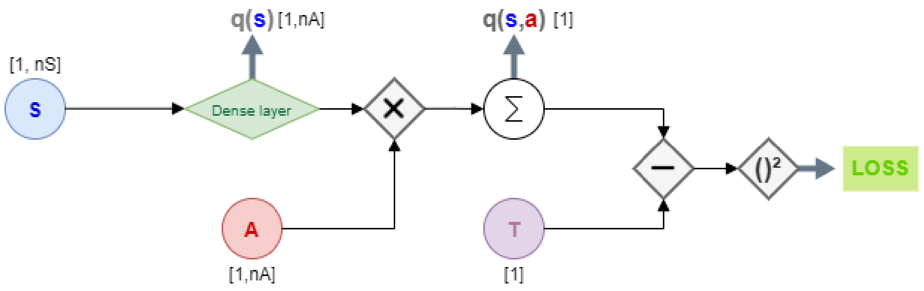

Fig. 1. neural network topology

5 Experiments

During these experiments the agent will learning through 1000 episodes.

5.1 Learning rate alteration

High learning rates causes drastic updates which leads to divergent behaviours. Thus the agent won't be capable of learning. However, too small learning rates require many updates to reach the optimal point. these facts can be easily seen in figure2. The highest total-rewards was with a relatively moderate learning rate.

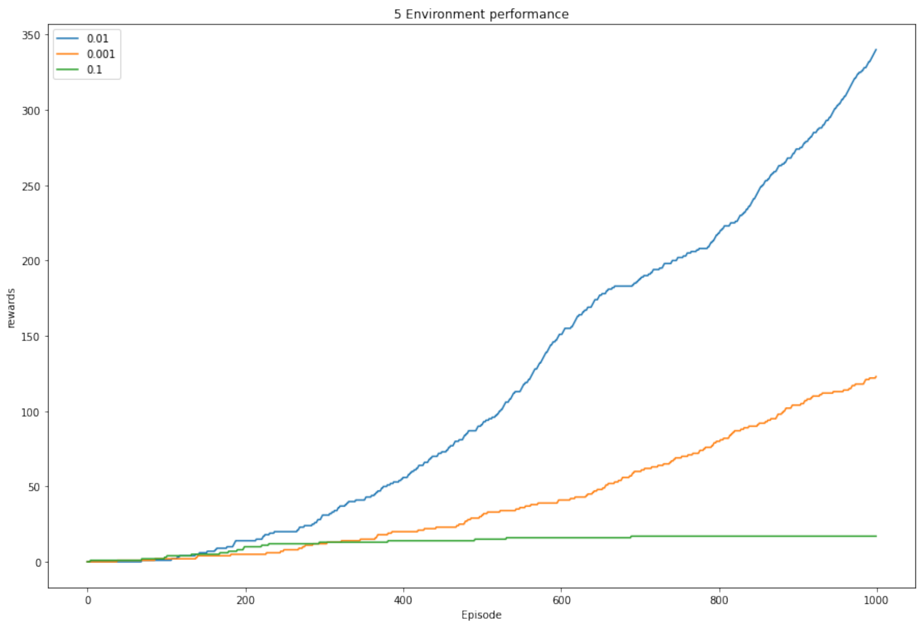

Fig. 2. Total rewards for different learning rates

5.2 Discount factor alteration

The discount factor quantifies how much importance given for future rewards. Gamma varies from 0 to 1. If it is closer to zero, the agent will tend to consider only immediate rewards. If it is closer to one, the agent will consider future rewards with greater weight, willing to delay the reward.

In this environment, having a higher discount rate give rise to a higher total-rewards.

5.3 Epsilon alteration

This parameter is introduced to balance the exploration of the full set of actions, and selecting actions from the Q-learning policy using the Q-table. Thus, Epsilon will represent the probability of prioritizing an exploratory action over a policy action.

This variable is decayed through time. Hence, as the agent learns a more optimal q-table at each episode, epsilon will be multiplied by a decay factor.

The effect of the epsilon decay factor is highlighted in figure4.

If the decay factor is low, epsilon will be near zero after few episodes, thus the agent won't be able to explore all the action space, and will keep choosing the greedy action from the randomly initialized Q-table.

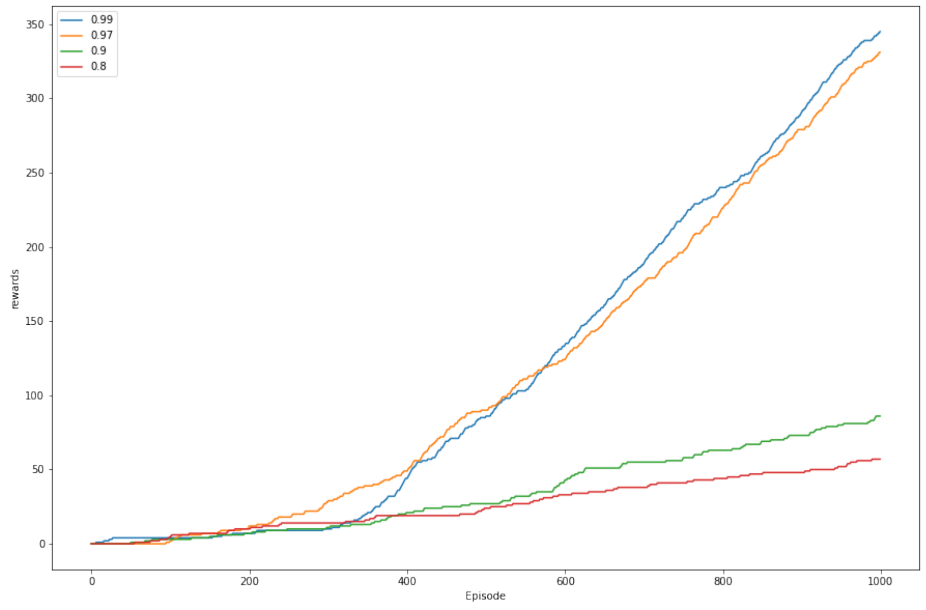

Fig. 3. Total rewards for different discount factors

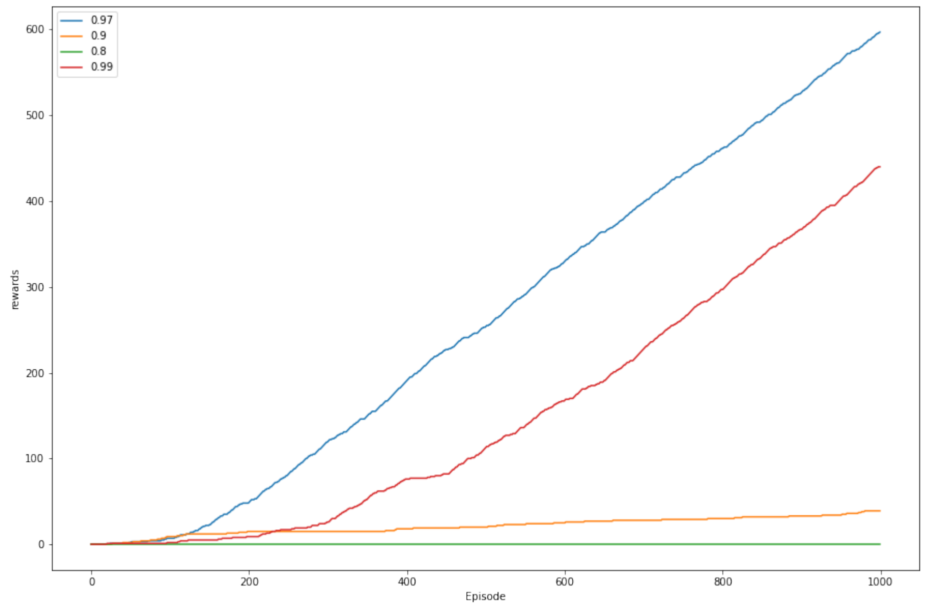

Fig. 4. Total rewards for different Epsilon decays

However, if it is too high, the probability of selecting the greedy action will stay minimal and the random choices will be more frequent. That's why an optimal decay factor should be set to ensure the maximum rewards.

6 Conclusion

In this report, multiple experiments were conducted to study the behaviour of a SARSA algorithm agent in the frozen-lake environment.

The stochastic criteria of this environment (the lake is slippery) made the task harder for the agent. However, with the right hyper-parameters it was able to get remarkable rewards.

This algorithms can be improved by a technique called experience replay which is a method that allows the agent the use old experiences to learn the optimal Q-table.

CODE : https://github.com/amine-oueslati/DRL/blob/main/FrozenlakeDeepRL.ipynb