Assignment report Reinforcement Learning

Ali Charara

Centralesupélec

Abstract. In this assignment, we implemented two agents to solve the Text Flappy bird environment. We will answer to the questions asked in the assignment. The implemented notebook can be found at https://github.com/ali-charara/assignment_rl

1 How did you choose the agents for the task ?

We first found a fairly simple policy to implement and tested TFB on it. The considered agent is a simple react agent and actually performs pretty well with few engineering underneath. We could try to improve it incrementally but we will dive deeper into Model free control agents which will implicitely learn a policy through experience.

We can model the environment of text flappy bird as a MDP where the states are the horizontal and vertical distance to the center of the next coming pipe (dx, dy), the actions are either Idle or Flap, the reward is 1 for each frame the bird survives and the final state is the death of the bird. This markov process is unknown, therefore we will need to improve the performances and learn policies in a model free setting. The two agents we implemented to this end are the Expected Sarsa agent as well as the Q-learning agent which are compatible model free control agents.

2 How are the two implemented agents different (e.g. sensitivity to parameters, convergence time, rewards, scores)?

Firstly, over all the agent training that were done, Q-learning training tend to be slightly faster than Expected Sarsa. Hence, Q-learning converges faster. Their sensitivity to parameters is high for both agents. We trained them for different values of epsilon, step size and discount. The plots are visible in Figure 1., 2. and 3.. We capped the score of one instance of flappy bird to a score of 10000 and averaged the performances over 10 runs for these plots and trained over 10000 episodes. The empirical means might be biased but still give us an idea on the sensitivity to different parameters. Overall, it looks difficult to compare the two agents in terms of performance with respect to different parameters. While Q-learning performs rather good for almost any discount value, Expected Sarsa only works for high discount values. This difference might be due to the

optimistic update of the Q-learning. On the other hand, in terms of exploration, Expected Sarsa is way less sensitive to the parameter which is essentially due to the fact that the updates are done by computing expectation over the policy and are therefore tailored to the exploration parameter which is not the case for Q-learning. Finally, Q-learning looks way less sensitive to step size than Expected Sarsa.

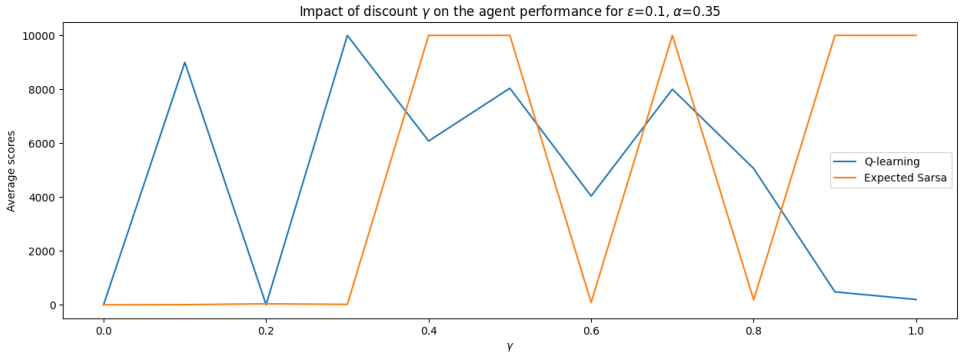

Fig. 1. discount γ impact

Fig. 2. exploration ϵ impact

Finally, the difference in the agents lies in the way their philosophy of action which can be seen on the value state functions:

Here we see, than unlike Q-learning chich doesn't make a difference between being above or under the pipe gap, Expected Sarsa values more getting under it, most probably because of the effect of gravity which makes entering the pipeline from above more difficult.

Fig. 3. step size α impact

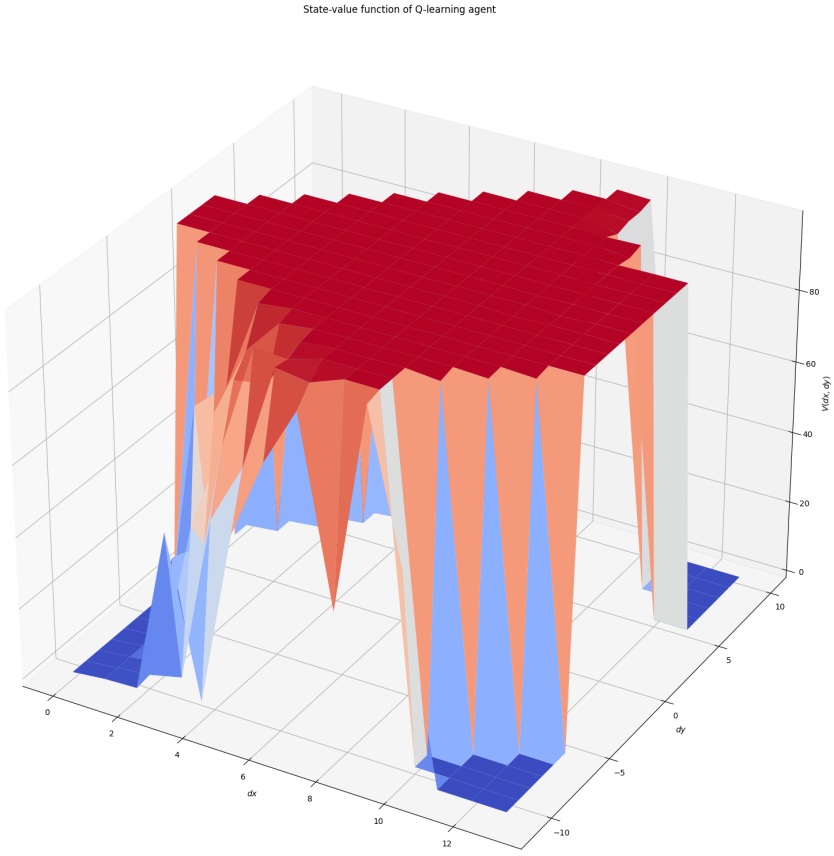

Fig. 4. $V(s)$ for Q-learning $\gamma = 0.99, \epsilon = 0.1, \alpha = 0.35$

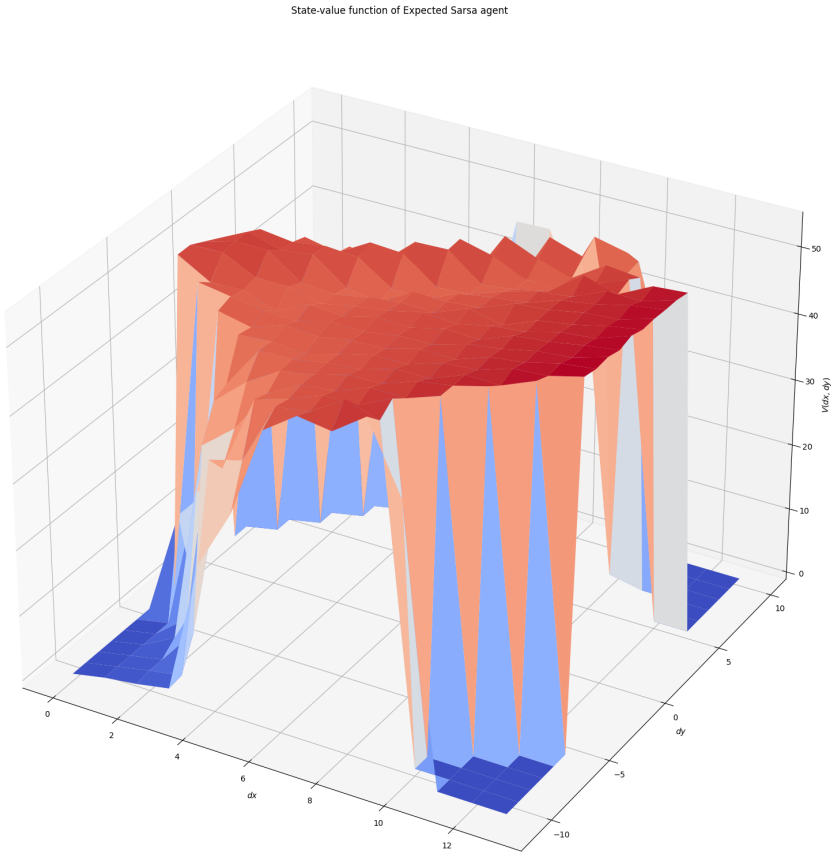

Fig. 5. $V(s)$ for Expected Sarsa $\gamma = 0.99, \epsilon = 0.1, \alpha = 0.35$

3 How are the two versions of the TFB environment different? What are the main limitations of using the same agent for the TextFlappyBird-screen-v0 ?

The only difference lies in the fact that for screen, the observation are not dx, dy but the render of the game as an array. The main limitations are the need to parse this array at each iteration to retrieve manually dx and dy which might computationally be too expensive to react online. Otherwise if the array is used as a state, the considered state space is way bigger than the no screen version which makes it not usable as is.

4 With an implementation of the original flappy bird game environment available, can the same agents be used?

According to the github, the observation sent by the environment are also dx, dy, the reward and the actions are the same. However, the screen size is way larger and dx, dy are potentially non integers. The state space is therefore way bigger than in our setting. Theoretically speaking, by rounding the observations, the agents should be able to learn on the original flappy bird environment. However, the training will anyway take way longer to run. In case it is too long to run, we might use value function approximation through function classes or approximators such as neural networks.