RL Agents for Text Flappy Bird

Alexandre Robin

CentraleSupélec

Abstract. This project explores the use of two reinforcement learning algorithms, Monte-Carlo and SARSA, to train agents for a simplified Flappy Bird game in two different environments. The agents are adapted to the different environments and their performance is compared based on their sensitivity to parameters, convergence time, rewards, and scores. It was found that SARSA(λ) agent performs better than Monte-Carlo Control in the `TextFlappyBird-v0` environment. It was also stated that trying to generalize an agent to both environment was not optimal and training specialized agents for each of them was a better strategy. Generalization to the real Flappy Bird is possible in when observation space correspond in the case of the `TextFlappyBird-v0`, when using the image of the game as observation space, the generalization would be too complicated.

Keywords: Reinforcement Learning · Flappy Bird · Monte-Carlo · SARSA.

1 Infrastructure

In this project, we investigate two simplified environments for the Flappy Bird game. The game's logic and graphics is the same for both environment but the observations returned to the agent differ. They are built as child of the OpenAI Gym Environment and match the Gym infrastructure.

1.1 States

The `TextFlappyBird-v0` environment yields simple numerical information about the game' state as observations, that is the horizontal distance to the next pipe and the difference between the player's y position and the next hole's y position.

The `TextFlappyBird-screen-v0` environment yields a simplified text representation of the screen, that is a matrix in which 1 corresponds to the bird and 2 corresponds to pipes, other values are set to 0.

Table 1 shows a basic set up we can work on. The baseline of our project is computed with a 20×15 game and then we try to project them to other configurations.

1.2 Actions

We represented an action a using $a \in 0, 1$. It is defined in our environment that 1 represents the "flap" action, and 0 represents the "idle" or "do nothing" action.

Table 1. Example of 10 × 10 environments at instanciation.

Environment	`TextFlappyBird-v0`	`TextFlappyBird-screen-v0`
Observation space	Tuple(Discrete(7), Discrete(12, start=-6)))	Box(0, 3, (10, 10), int32)
Action space	Discrete(2)	Discrete(2)
Initial state	(13, 1)	$\begin{bmatrix} 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 2&2&0&0&0&0&2&2&2&2 \end{bmatrix}$

1.3 Reward

The reward considered is very simple and simply yields 1 at each step the agent remains alive. As long as we stay alive, the cumulative reward increases.

2 Agents

2.1 Baseline

For the baseline algorithm, we simply apply a random policy. The random policy picks an action or the other with same probability.

2.2 Monte-Carlo

The first agent we impemented is a Monte-Carlo based agent. We implemented a Monte-Carlo Control with a soft policy and an arbitrary tie breaking. Monte-Carlo Control learns from complete episodes of experience. It updates action values based on the total return observed after the entire episode.

2.3 SARSA(λ)

The second agent we considered is a Sarsa(λ) agent as described in section 12.7 of the Reinforcement Learning, An introduction by Sutton and barto. SARSA(λ) is a TD(λ) method, which combines elements of both TD learning and MC methods. It updates action values based on a bootstrapping approach, incorporating both the reward received and the estimated value of the next state-action pair.

2.4 Convergence of the agents

Monte-Carlo Control tends to converge to the optimal policy as more episodes are sampled. However, since it updates based on complete episodes, it may take longer to converge, especially in environments with long episodes.

SARSA(λ) typically converges faster than MC Control because it updates values after each step, rather than waiting until the end of the episode.

2.5 Sensitivity

Monte-Carlo Control is generally less sensitive to its parameters compared to SARSA(λ). The most important parameter in MC Control is the exploration-exploitation trade-off. Tuning ϵ can affect the rate of exploration and exploitation, but it doesn't have a significant impact on the overall performance of MC Control.

SARSA(λ) is more sensitive to its parameters, particularly the λ parameter which determines the degree of eligibility trace used in the action values update. Higher λ value includes more distant state-action pairs in the update process, that can lead to faster convergence but also increase the variance of the estimates.

3 Results

Standard training has been done on 20000 games of `TextFlappyBird-v0` with height 15, width 20 and pipe gap 4. For each game the max reward (ie: max steps) is 5000. Discount factor is set to 0.9. The epsilon greedy strategy starts with a parameter epsilon of 1.0 that is decayed until 0.1 at a rate 0.9999. A simple exploration of parameters is realized for each agent and detailed in relevant subsections.

3.1 Baseline

When testing the results obtained with the baseline model yield an average reward of 11.5 corresonding to a score in the game of 1, signifying that the average random agent goes through the first door only. Figure 1 shows the results of the baseline.

3.2 Monte Carlo Control

Along standard parameters described at the beginning of the section, standard learning rate for Monte Carlo Control trainig is set to 0.2. Parameter exploration yields along discount factors $0.9, 0.95, 0.99$, learning rates $0.02, 0.01, 0.001$, max step $100, 1000, 5000$ and number of episodes $1000, 5000, 10000, 20000$, that best parameter combination was discount factor 0.95, learning rate 0.01, max steps 5000 and number of episodes 20000. The top results are found in Table 2. Figure 2 shows the training rewards and epsilon evolution, Figure 3 shows the best policy found and Figure 4 the state-action values for the policy. On test over 10000 games with max step 5000 the policy had an average reward of 4595.

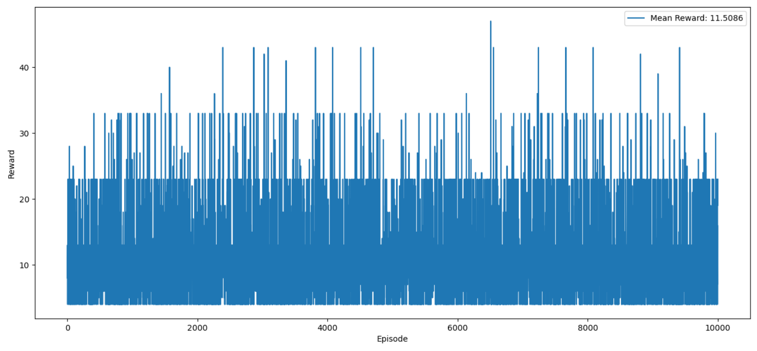

Fig. 1. Baseline rewards on 10000 tests.

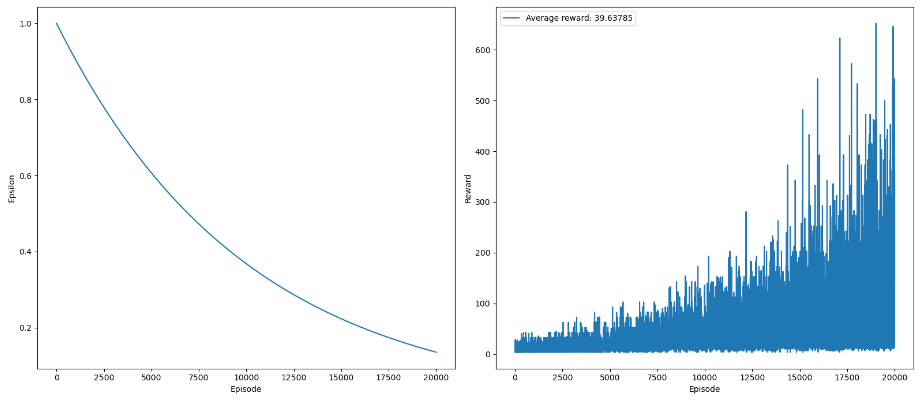

Fig. 2. Monte Carlo Control training setting on 20000 tests with optimal parameters.

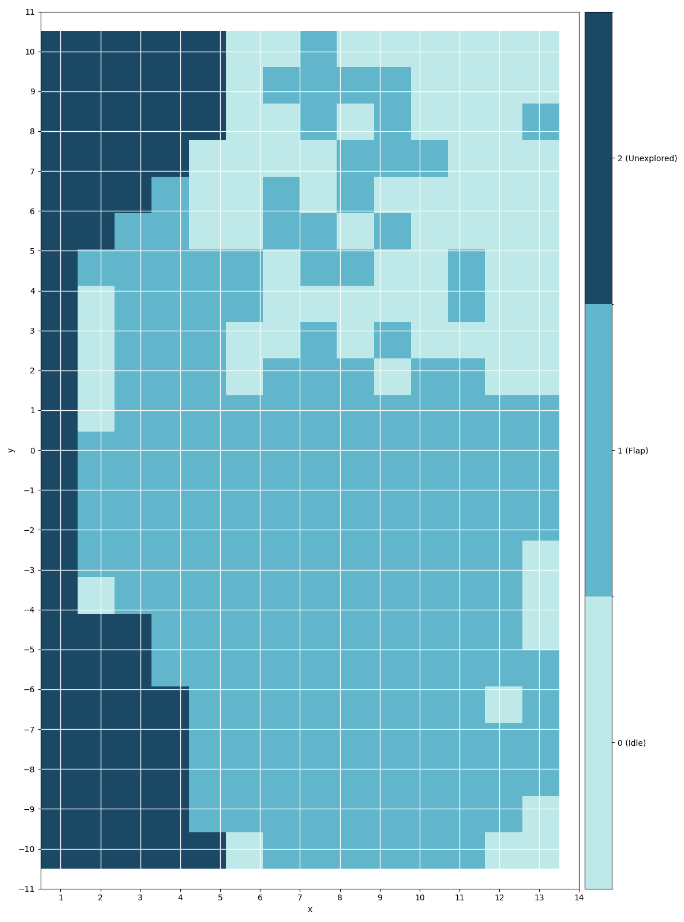

Fig. 3. Monte Carlo Control policy.

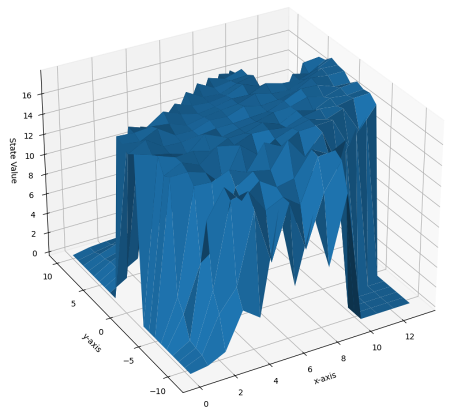

Fig. 4. Monte Carlo Control state-action values.

Discount factor	Learning rate	Max. steps	N. episode	Avg. reward
0.95	0.01	5000	20000	40.74
0.90	0.01	1000	20000	39.47
0.95	0.02	1000	20000	39.43
0.95	0.02	5000	20000	39.09
0.90	0.01	5000	20000	38.99

Table 2. Top results on training for Monte Carlo Control

Environments change We test the policy on various environments with variations of the standard parameters of 25%. Results are gathered in Table 3. We find that the agent improves when increasing pipe gaps, but deteriorates when decreasing pipe gaps. It keeps performing well if we reduce height by 25% but increase pipe gap by 25%. When augmenting width by 25% it never performs well. However, it keeps performing well when we reduce height and width together by 25% for same pipe gap or bigger. Figure 5 shows some results of this exploration.

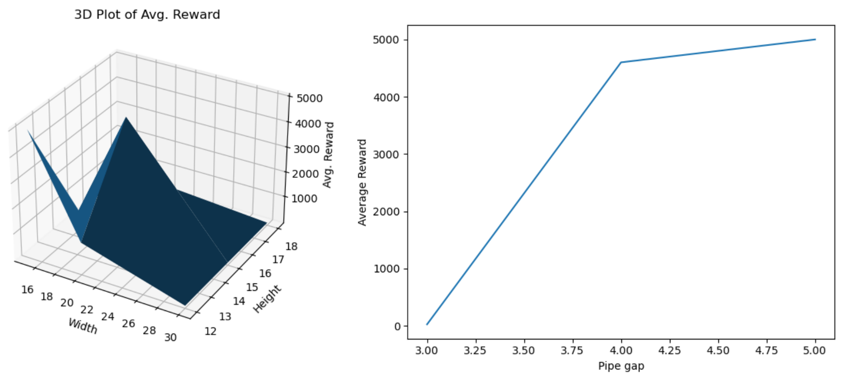

Fig. 5. Results of varying environment parameters.

3.3 Sarsa Lambda

We explore the same parameter space as for Monte-Carlo Control to which we add the λ parameter space that is $0.25, 0.5, 0.75$. The best parameter combination was discount factor 0.99, learning rate 0.01, and lambda.0.75, letting us suppose that the agent worked better for a big λ. The top results are found in Table 4. Figure 6 shows the training rewards and epsilon evolution, Figure 7 shows the best policy found and Figure 8 the state-action values for the policy. On test over 10000 games with max step 5000 the policy had the maximum reward each time.

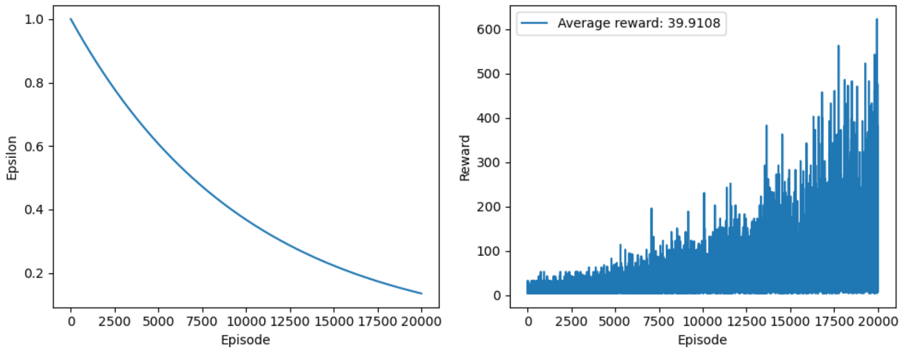

Fig. 6. SARSA(λ) training setting on 20000 tests with optimal parameters.

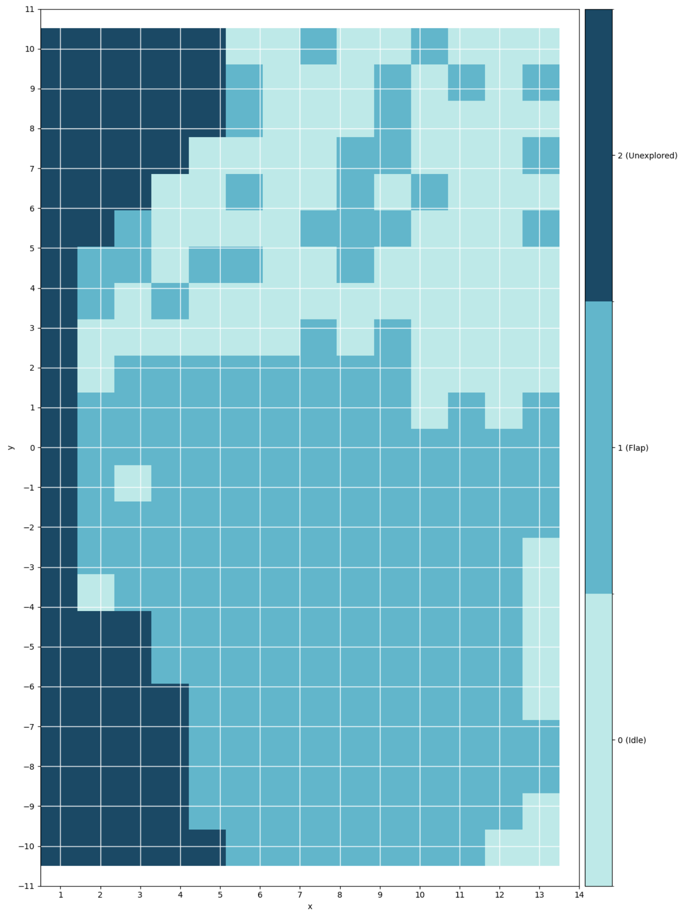

Fig. 7. SARSA(λ) policy.

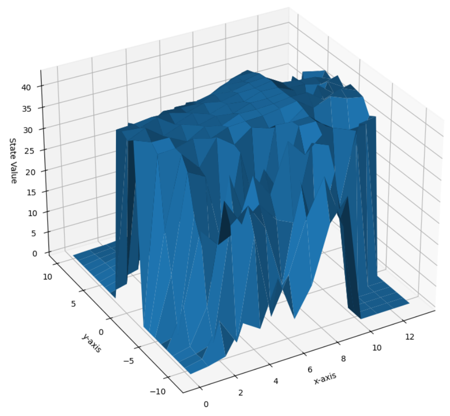

Fig. 8. SARSA(λ) state-action values.

Height	Width	Pipe Gap	Avg Reward
12	15	3	22.0
12	15	4	5000.0
12	15	5	5000.0
12	20	3	28.0
12	20	4	1194.0
12	20	5	5000.0
12	30	3	3.0
12	30	4	3.0
12	30	5	3.0
15	15	3	21.0
15	15	4	278.0
15	15	5	938.0
15	20	3	26.0
15	20	4	4601.0
15	20	5	5000.0
15	30	3	4.0
15	30	4	4.0
15	30	5	4.0
18	15	3	22.0
18	15	4	90.0
18	15	5	107.0
18	20	3	22.0
18	20	4	318.0
18	20	5	1307.0
18	30	3	4.0
18	30	4	4.0
18	30	5	4.0

Table 3. Environment parameter sweep result for Monte-Carlo Control with optimal parameters

Environments change We test the policy on various environments with variations of the standard parameters of 25%, as in Monte-Carlo Control section. Results are gathered in Table 5. We find that the agent improves when increasing pipe gaps, but deteriorates when decreasing pipe gaps. It keeps performing well if we reduce height by 25% even without increasing pipe gap. When augmenting width by 25% it never performs well. However, it keeps performing well when we reduce height and width together by 25% for same pipe gap or bigger. Figure 9 shows some results of this exploration. Overall we find that Sarsa(λ) is a bit more robust to environment change than Monte-Carlo Control.

4 Extension to another environment and real Flappy Bird

We cannot use the same agent for `TextFlappyBird-v0` and for `TextFlappyBird-screen-v0` as the observation varies too much between them. Furthermore the agents may

Discount factor	Learning rate	λ	Avg. Reward
0.99	0.01	0.75	15.0462
0.99	0.02	0.75	14.9884
0.95	0.02	0.75	14.8874
0.99	0.02	0.50	14.8538
0.95	0.01	0.75	14.6176

Table 4. Top results on training for Sarsa(λ) for 5000 episodes with max steps 5000.

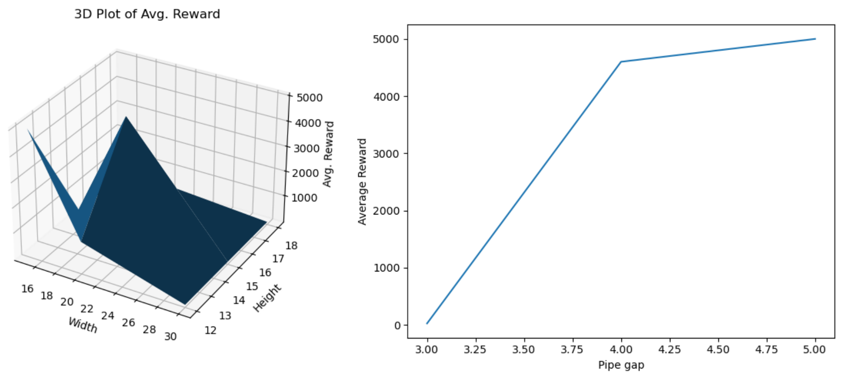

Fig. 9. Results of varying environment parameters.

need to rely on different strategies to perform in each of the environments. It therefore would be more optimal to trian different agents to solve the different versions of the environment. Still, considering the same observation space, an agent performing in the `TextFlappyBird-v0` environment can easily be extended to the real flappy bird game. However it would not for a Flappy Bird game in which the observation corresponds to the image of the game.

Height	Width	Pipe Gap	Avg Reward
12	15	3	20.0
12	15	4	5000.0
12	15	5	5000.0
12	20	3	20.0
12	20	4	5000.0
12	20	5	5000.0
12	30	3	3.0
12	30	4	3.0
12	30	5	3.0
15	15	3	20.0
15	15	4	1130.0
15	15	5	5000.0
15	20	3	20.0
15	20	4	5000.0
15	20	5	5000.0
15	30	3	4.0
15	30	4	4.0
15	30	5	4.0
18	15	3	20.0
18	15	4	137.0
18	15	5	216.0
18	20	3	16.0
18	20	4	462.0
18	20	5	1245.0
18	30	3	4.0
18	30	4	4.0
18	30	5	4.0

Table 5. Environment parameter sweep result for SARSA(λ) with optimal parameters