Reinforcement Learning Flappy Bird Assignment

Nicolas Bourriez

CentraleSupélec

Abstract. This project explores the use of two reinforcement learning algorithms, Expected SARSA and Deep Q Learning, to train agents for the Flappy Bird game in two different environments: TextFlappyBird-v0 and TextFlappyBird-screen-v0. 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 DeepQLearning performed better in the TextFlappyBird-screen-v0 environment, possibly due to its ability to extract relevant features from complex observations and adapt to new environments more effectively. The limitations of using the same agent for both environments were also discussed. Overall, this project provides insights into the application of reinforcement learning to complex game environments and highlights the importance of algorithm selection and adaptation to achieve optimal performance.

Keywords: Reinforcement Learning · Flappy Bird · Expected SARSA · Deep Q Learning

1 Choice of Agents

For this Flappy Bird project, I wanted to compare two very *different* agents, eventhough the problem is "simple" in its form and many "similar" algorithms can be used.

Thus I first chose **Expected SARSA** as my first agent since it can both deal with *discrete* and *continuous* state and action spaces (anticipating the potential use of the `TextFlappyBird-screen-v0` environment). Additionally, one of the main reasons is also that Expected SARSA is known to be more **robust** and **safer** than other algorithms, such as Q-learning, in the presence of *noisy* and *incomplete* information such as ours (i.e. we do not see the full screen here).

The second agent could have been quite similar, such as Q-Learning, but I wanted to step aside and use a **model-free function approximator**. Hence the choice of model-free **Deep Q Learning**. Even if our `TextFlappyBird-simple-v0` environment is not high-dimensional at all, I wanted to also first anctipate in case of high-dimensional state spaces such as the full screen, and second to check if the *"If you can do more, you can do less"* idea holds in this context.

2 Comparison of Agents

The two implemented agents, Expected SARSA and Deep Q Learning, differ in **several important ways**.

First, in terms of **sensitivity to parameters**, Expected SARSA is generally *far less sensitive* than Deep Q Learning. This means that small changes to the hyperparameters, such as the learning rate or exploration rate, are less likely to have a significant impact on the performance of Expected SARSA than on Deep Q Learning. On the other hand, Deep Q Learning is **extremely** sensitive to such parameters, notably regarding the `tau` to update the Target network, and the`replay_buffer_size` for experience replay (see Appendix).

Second, Expected SARSA generally has a **faster convergence time** than Deep Q Learning. To interpret this, we can think that it is because Expected SARSA is a *one-step* algorithm, which means that it updates its Q-values based on a single transition from the current state to the next state. In contrast, Deep Q Learning is a *multi-step* algorithm that updates its Q-values based on a sequence of transitions, batched from the Replay Buffer. This can lead to slower convergence times for Deep Q Learning.

Third, the rewards and scores obtained by the two agents can vary depending on the task and environment. In general, Expected SARSA tends to perform well on tasks that require more exploration and exploitation, while Deep Q Learning tends to perform well on tasks that require more complex decision-making. In the case of Flappy Bird Gym, both agents were able to achieve high scores, but the specific scores obtained varied a lot depending on the implementation and hyperparameters used (see Appendix)

Overall, the choice between Expected SARSA and Deep Q Learning depends on the specific requirements of the task and the environment, as well as the available computational resources and time constraints. We will also see in another section that *in practice* while playing, the two algorithms do not yield same results.

3 Comparison of Environments

3.1 Recap of differences

There are two available versions of the TextFlappyBird environment, `TextFlappyBird-screen-v0` and `TextFlappyBird-v0`, which differ in the information provided in the observation space. `TextFlappyBird-screen-v0` returns the complete screen render of the game as an observation, while `TextFlappyBird-v0` returns the distance of the player from the center of the closest upcoming pipe gap along the x and y axes.

Algorithm	Num Episodes	Average Score	Average Reward Sum
Expected SARSA	10 000	2.728	38.7
Expected SARSA	30 000	9.137	103.048
Expected SARSA	50 000	9.801	110.091

Algorithm	Num Episodes	Decay	Tau	Average Score	Average Reward Sum
Deep Q Learning	1000	1000	0.1	10.598	117.24
Deep Q Learning	1000	1000	0.01	9.905	110.231
Deep Q Learning	1000	1	0.005	6.751	77.852
Deep Q Learning	1000	1000	0.005	7.1	81.59
Deep Q Learning	500	1000	0.005	2.118	32.246

3.2 Limitation of using same agent

The main limitation of using the same agent for both versions of the TFB environment is that the observation space is significantly different between the two versions. `TextFlappyBird-screen-v0` provides much more visual information about the game state, including the position and movement of obstacles and the bird, whereas `TextFlappyBird-v0` only provides information about the distance to the nearest pipe gap. This means that an agent trained on one version of the environment may not be able to effectively generalize to the other version, as the **state space** and **action space** are **different**.

Furthermore, the two versions of the environment may require **different strategies and actions** to achieve a high score. For example, an agent trained on `TextFlappyBird-screen-v0` may learn to use visual cues to avoid obstacles and navigate through the game, while an agent trained on `TextFlappyBird-v0` may need to focus on accurately predicting the distance to the nearest pipe gap and taking actions accordingly. This means that using the same agent for both versions of the environment may not be optimal, as it may **not be able to effectively *adapt*** to the differences in the observation space and task requirements.

In summary, while both versions of the TFB environment present similar challenges and goals, the **differences in the observation space** and task requirements mean that using the same agent for both may not be optimal. Instead, it may be more effective to train separate agents for each version of the environment, or to use a more generalizable approach that can adapt to the differences in the observation space and task requirements.

4 Usecase of Screen Environment

The Expected SARSA and Deep Q Learning agents can be used with the original Flappy Bird game environment (`TextFlappyBird-v0`). This is because the environment still presents the same basic challenge of avoiding obstacles and achieving a high score by navigating through the game. However, it's important to note that the observation space is different in the original game compared to the modified `TextFlappyBird-screen-v0` environment, as the original game does not provide a pixel-level screen render. Therefore, the agents may need to be adapted or retrained to effectively utilize the available observation space and achieve high scores in the original game:

- **Expected SARSA:**
  to use Expected SARSA with the `TextFlappyBird-screen-v0` environment, the agent will need to be adapted to work with the pixel-level screen render that is provided as an observation. This may involve developing new techniques for feature extraction and decision making based on the visual information, such as identifying the position of the bird and the obstacles

on the screen. Additionally, the agent may need to be retrained on a larger dataset, as the pixel-level screen render provides more information compared to the limited observation space available in the original Flappy Bird game environment.

– **Deep Q Learning:**
similarly, Deep Q Learning will require some adaptation to work with the `TextFlappyBird-screen-v0` environment. The main challenge with this agent is the larger observation space, which includes the pixel-level screen render. This means that the agent may need to learn more abstract features to effectively navigate through the game and identify the optimal actions to take. One option may be to include **convolutional layers** in the `QNetwork` class to account for this screen render and subtract meaningful information. Additionally, the agent may need to be trained on a larger number of episodes to ensure that it has enough examples to learn the necessary features and make accurate predictions.

5 In Practice

I created a pipe in which the trained agent is used to play directly in the terminal.

Testing the two trained agents, one odd fact appears: *it seems that when playing thee game with the DQN agent trained, it performs far worse than with ExpectedSARSA even though DQN has higher score during training.*

One main reason I found is in the **unstability** of the DQN during training, making it **less robust** over time. Since DQN is known to be sensitive to hyperparameters, such as the *learning rate*, *discount factor* but mainly *tau* and *replay size*, the agent may not generalize well to new environments, leading to poor performance. Additionally, DQN requires a far larger number of episodes compared to Expected SARSA, which could make it more prone to **overfitting**.

Expected SARSA, on the other hand, is less sensitive to hyperparameters and can often achieve **good performance** with a *smaller dataset*.

In summary, the poor performance of the DQN agent on the TextFlappyBird-v0 environment may be due to **overfitting** and sensitivity to hyperparameters. Expected SARSA may be better suited to this environment due to its ability to extract relevant features, its robustness and adapt to new environments more effectively.

6 Appendix

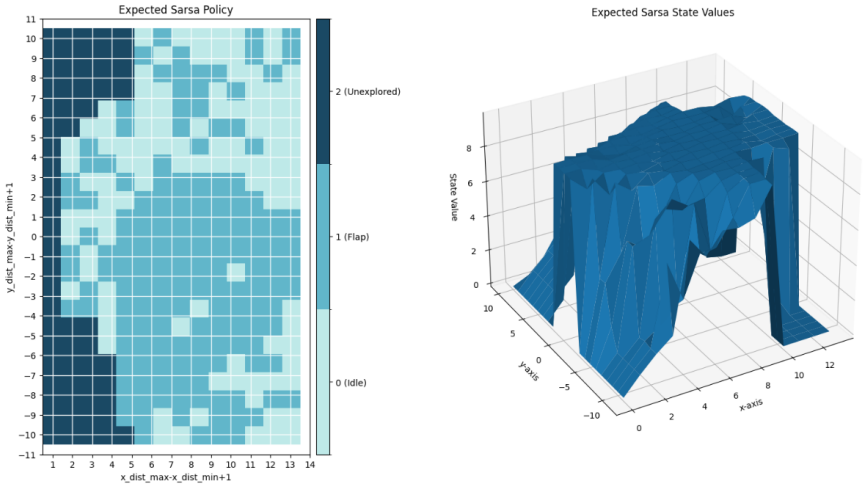

Fig. 1. The Expected SARSA Policy and State Value function over 10 000 episodes

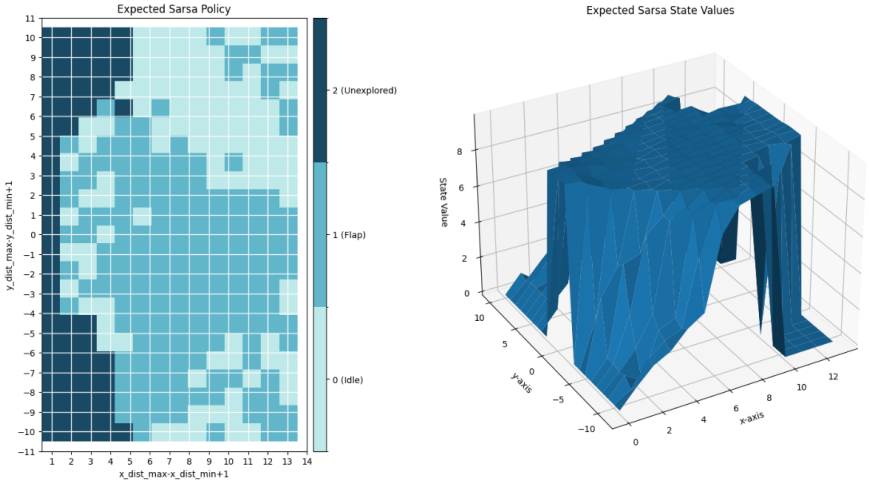

Fig. 2. The Expected SARSA Policy and State Value function over 30 000 episodes

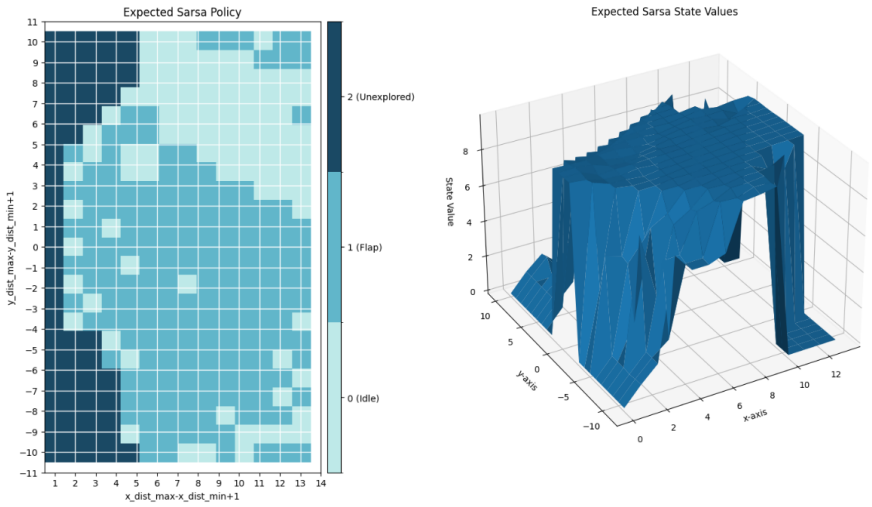

Fig. 3. The Expected SARSA Policy and State Value function over 50 000 episodes

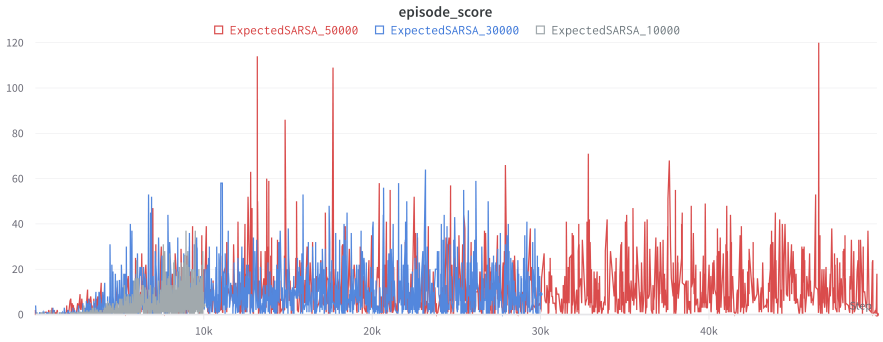

Fig. 4. The different episodes scores of Expected SARSA algorithm

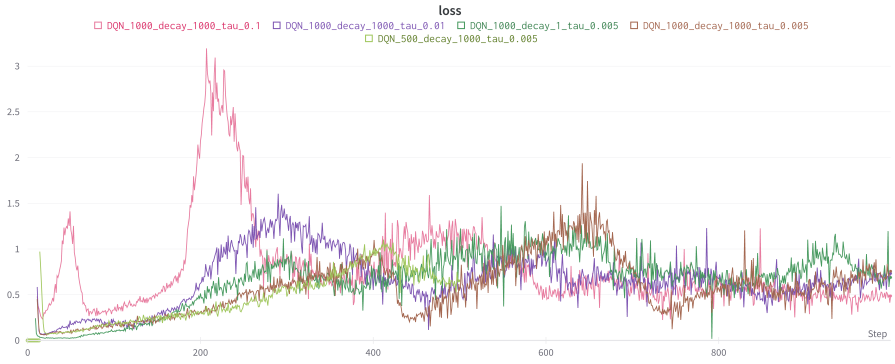

Fig. 5. The very sparse losses according to the hyperparameters for the Deep Q learning