Reinforcement Learning for FlappyBird

Roja Immanni

Introduction

FlappyBird is a mobile game introduced in 2013 with a bird as the protagonist. The player controls the bird trying to make him fly between the green vertical pipes without hitting them. The player can make the bird fly by tapping on the screen or not do anything to keep him flying between the pipes. The player loses when the bird touches the pipe. In this project, we use reinforcement learning techniques to train the bird to keep flying between the pipes without losing for as long as possible.

Framework:

Based on the above rules, the RL problem for this game can be formulated in the following way:

- Environment: Flappybird's game space
- Agent: Agent is the bird who decides either to do nothing or jump
- States: Bird's vertical and horizontal distance from the next pipe and its speed
- Actions: Actions would be either to do nothing or jump
- Rewards: positive reward(+1) if the bird is still alive and negative reward(-1000) if it hits a pipe

Methods:

We have tried the on-policy(SARSA) and off-policy technique(Q-Learning) with the epsilon-greedy approach and compared it with the baseline model

Baseline: For the baseline model, we picked an agent that picks the next action randomly. The actions space contains (0, 1) where 0 stands for jump, and 1 stands for doing nothing. Random agent picks each of these actions with probability 0.5.

Setup:

The environment created for the flappybird captures its state 30 times every second. The RL agent can decide what action to take at each of these states based on the policy that they have chosen. At each step, the agent's state is captured by its horizontal and vertical distance from the next pipe and its speed.

For every extra state that the bird survives, he gets an extra +1 point added to the score. An iteration ends when the bird hits a green pole and dies. So while evaluating the trained model, the maximum score among 100 iterations is captured.

Results:

The figure below shows the maximum score by different models when trained for 40,000 iterations. Clearly we have better results with both the models compared to the baseline model i.e the random agent. There is a huge performance difference between SARSA and Q-learning. Q-learning algorithm has achieved higher scores in a lesser number of iterations. Q-Learning performed better over SARSA which would be because Q-learning is an off-policy that uses the optimal policy directly estimating the optimal policy quicker than SARSA which is on-policy learning and it has to go through all the combinations of the actions at every state to find the best policy.

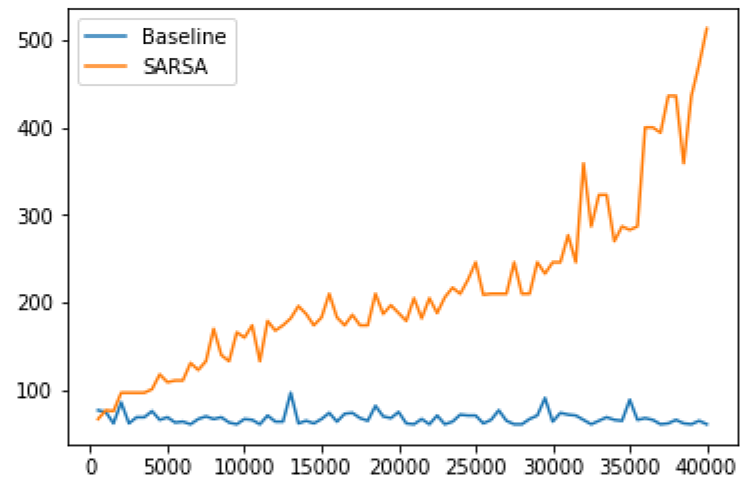

Fig1: Baseline versus SARSA model performance over iterations

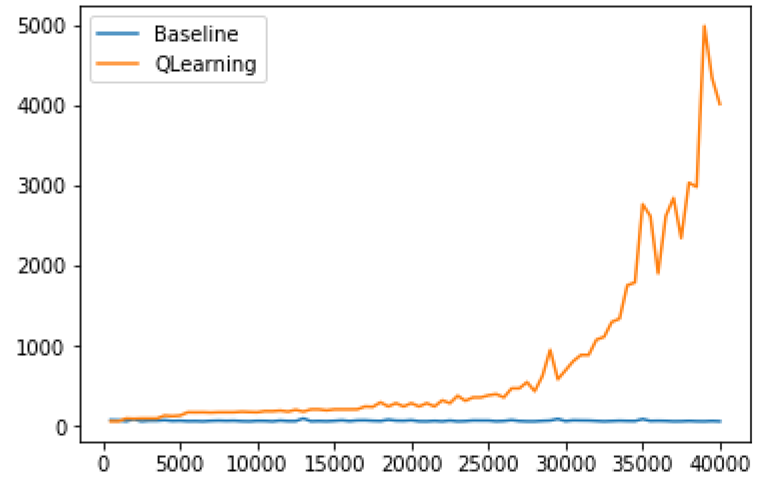

Fig2: Baseline versus Q-Learning model performance over iterations

Future Work:

First, the only hyperparameter tuning done was for the epsilon value to set the level of exploration. We can adjust more hyperparameters for the model such as the learning rate and discount rate.
As we noticed in the graph, it has taken a lot of iterations to see progress in the performance. We can reduce the number of distinct states by rounding off the state values to reduce the number of unique states and help the algorithm learn faster and improve performance.