Combining Q-Learning with Artificial Neural Networks in Flappy Bird
Project Report of COMP 599

Jianan Yue
McGill University
Montreal, Quebec H3A0C3
Email: jianan.yue@mail.mcgill.ca

Yulin Shi
McGill University
Montreal, Quebec H3A0C3
Email: yulin.shi@mail.mcgill.ca

***Abstract*—For the flappy bird game, a typical delay control problem, we propose using the reinforcement learning algorithms to learn the control policy. Specifically, we use the Bellman equation to propagate the end-point reward to the entire state space; use model free Q-function to search optimal policy; and, to deal with the curse of dimension, use the neural network to parametrise the approximate function of the optimal controller. The Flappy Bird environment is developed in the JavaScript. The open source machine learning library Convnetjs is included to train the neural network. It is validated that the neural network Q-learning is proper for the learning problem. It is also observed that some strategies emerged after training the simple neural network.**

1. Introduction

1.1. Flappy Bird Problem

The well-known game Flappy bird, in a functional perspective of view, is a simulation world consists with a part of deterministic dynamic: the acceleration and a part of stochastic dynamic: the position of the gap. By observing the location of the path and estimating the trajectory of the bird in a continuous time, the players selects among two input commands. The object is to go through as many gaps as possible while avoiding colliding with the pipes. The game can be described in discrete time as a Markov process.

$$s' \sim \varepsilon(s, a) \tag{1}$$

where s is the current state, a is the current action, s' is the next state. Symbol $\sim$ mean the value of s' is subject to a distribution.

In this view, it is feasible to use an artificial agent following a policy (2) such as the ϵ-greedy policy to replace human as the player to play the Flappy Bird game. [**?**]

$$a = \pi(s) \tag{2}$$

1.2. Reinforcement Learning

One difficulty of learning the Flappy Bird problem is that the leaner agent does not perceive a prompt reward after taking an action. Rather, they get the reward at the end of an episode which is defined as the entire state-action trajectory of a round of play. At the end of an episode, the reward is proportional to the number of pipes the bird goes through. The agent is trained to learn a policy that maximize the overall reward. However, the strong delayed reward in the Flappy Bird problem invalids the classical machine learning algorithms such as the regression and classification which assumes that the samples are independent. For example, the agent does not perceive any reward or penalty when it does not choose the jump action and stays far away from the gap, even if these actions and states have a strong negative influence on the subsequent rewards. Therefore, we need an index that reflect the expected utility of a state, other than the instantaneous reward. We call the index as the value function

$$V^{\pi}(s) = E_{\pi}\left\{R_t | s_t = s\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\} \tag{3}$$

where $0 < \gamma \leq 0$ is the discount factor.

For the Flappy Bird problem as a Markov process, the definition of the (3) can be reformulated as the Bellman equation (4) and solved using episode samples iteratively.

$$V^{\pi}(s) = E_{\pi}\left\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s\right\} \tag{4}$$

1.3. Q-learning

Another issue arising is that while the value function alone does not tell the agent how to act without a model (5) which can be very complex in real environments and usually suffers greatly from poor accuracy. [**?**]

$$\hat{s}' = \hat{\varepsilon}(s, a) \tag{5}$$

Since the Flappy Bird problem is basically a control problem, it is worthy of considering a model-free approach. Here we first define a Q function with respect to state and action

$$Q^{\pi}(s, a) = E_{\pi}\left\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s, a_t = a\right\} \tag{6}$$

Given the value of a Q function, it is possible to generate an optimal policy without a model

$$\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}} Q(s,a) \tag{7}$$

There are basically two approaches of learning the Q function given episode samples: off-policy (8) where actions are chosen greedily and on-policy (SARSA) (9) where actions are chosen based on the policy.

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \tag{8}$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma Q(\pi(s,a),a') - Q(s,a) \right) \tag{9}$$

2. Conclusion

The conclusion goes here.

Acknowledgments

The authors would like to thank...