Deep Reinforcement Learning on Car Racing Game

Jiahao Zhang, Minghe Ren, Weixuan Jiang, Zhonghao Guo
{jiahaozh, sawyermh, jwx0728, gzh1994}@bu.edu

Figure 1. Illustrate your task and/or approach

1. Task

Our project will explore the algorithms of reinforcement learning (RL) used in game playing. We intend to study several RL algorithms and implement them in a car racing game. The RL methods include, but not limited to, DQN, Double DQN(DDQN), and Dueling DDQN. After the RL methods been implemented, we will make comparison between these results and analyze the characteristics of each method.

2. Related Work

A well-known success story of reinforcement learning on games is *Playing Atari with Deep Reinforcement Learning*[1]. It introduced the Deep Q-learning, a new reinforcement learning which was for the first time that researchers successfully learned control policies directly from high-dimensional sensory input using reinforcement learning with deep learning model[2]. However, there was a problem that using the same parameters (weights) for estimating the target and the Q value results in a big correlation between the TD target and the parameters (w) we are changing. The idea of fixed Q-targets solved the problem.

Later in 2016, two modified version DQN, Double DQN and Dueling DQN were proposed in [3] and [4]. Double DQNs was introduced by Hado van Hasselt. This method handles the problem of the overestimation of Q-values and, as a consequence, helps us train faster and have more stable learning. While for dueling-DQN(aka DQN), we decompose Q(s,a) as the sum of V(s)(the value of being at that state) and A(s,a) (the advantage of taking that action at that state). And then we combine these two streams through a special aggregation layer to get an estimate of Q(s,a). By decoupling the estimation, our DDQN can learn which states are (or are not) valuable without having to learn the effect of each action at each state.

3. Approach

Introduce DQN DDQN (Dueling DQN)

We are going to implement three algorithms of RL in our application. They're DQN, Double DQN, and Dueling DQN. We will use tools like Tensorflow, OpenAI as our training and experiment platforms. We will write the source code of these algorithms and apply them on a existing game from OpenAI Gym.

Popular methods of RL can be roughly divided into to two categories: DQN and Policy Gradients. Although there are other algorithms that intend to improve the performance of both, we chose to focus on DQN and its subsidiaries as a startpoint of our project.

Deep Q Learning (DQN) adopts its idea from Q-learning. It can handle more complicate problems than Q-learning and also with less space (memory) complexity. DQN utilizes a Neural Network to estimate the Q-value function. Usually the input for the network is the current state, while the output is the corresponding Q-value for each of the action [5].

Double Deep Q Learning (Double DQN) intends to solve the overestimating issue when approximating action values in DQN, which cause slow convergence. Here, two networks are used to decouple the action selection from the Q value function.

Dueling Deep Q Learning (Dueling DQN) decomposes the Q-value function into one for the state value function and one for the state-dependent action advantage function [4]. This architecture usually helps to accelerate the training process and produce better performance.

4. Dataset and Metric

Reinforcement learning describes the set of learning problems where an agent must take actions in an environment in order to maximize some defined reward function.

Unlike supervised deep learning, large amounts of labeled data with the correct input output pairs are not explicitly presented.

In our car racing game, we don't have existing dataset. We imagine to set up an interesting game environment where a racing car can autodrive along the road. And it can not hit on the roadblocks. Here, agent is the racing car. We will track its actions and document corresponding rewards to construct its own dataset.

After searching into open source Open AI gym library https://github.com/openai/gym, we can learn from that to establish our own game environment.

Preliminarily, we can evaluate the results by documenting its moving coordinates and measuring the probability of driving out of road or hitting on roadblocks. Later on, we will have more detailed metrics to do evaluation.

5. Proposed Timeline and Roles

Each team mate should be assigned some non-trivial coding task.

Task	Deadline	Lead
Implement DQN	10/25/2018	Minghe Ren
Run test on DQN	10/31/2018	Weixuan JIang
Implement DDQN	11/25/2018	Zhonghao Guo
Run test on DDQN	11/31/2018	Jiahao Zhang
Prepare report and presentation	12/10/2018	all

References

1) Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning[J]. arXiv preprint arXiv:1312.5602, 2013.
2) Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529.
3) Van Hasselt H, Guez A, Silver D. Deep Reinforcement Learning with Double Q-Learning[C]//AAAI. 2016, 2: 5.
4) Wang Z, Schaul T, Hessel M, et al. Dueling network architectures for deep reinforcement learning[J]. arXiv preprint arXiv:1511.06581, 2015.
5) Steeve Huang, Introduction to Various Reinforcement Learning Algorithms. Part I (Q-Learning, SARSA, DQN, DDPG), https://towardsdatascience.com/@huangkh19951228

$$Q_t = r_t + \gamma \max_{a'} Q(S_{t+1}, a'; \theta)$$