Valparaiso University
ValpoScholar

Graduate Academic Symposium | Graduate School

Spring 4-29-2021

Framework for Deep Reinforcement Learning Experimentation

HarishGupta Lingam
harishgupta.lingam@valpo.edu

Follow this and additional works at: https://scholar.valpo.edu/gas

Recommended Citation

Lingam, HarishGupta, "Framework for Deep Reinforcement Learning Experimentation" (2021). *Graduate Academic Symposium*. 82.
https://scholar.valpo.edu/gas/82

This Oral Presentation is brought to you for free and open access by the Graduate School at ValpoScholar. It has been accepted for inclusion in Graduate Academic Symposium by an authorized administrator of ValpoScholar. For more information, please contact a ValpoScholar staff member at scholar@valpo.edu.

Framework for Deep Reinforcement Learning Experimentation

Harish Gupta Lingam, Spring 2021

In recent years deep learning obtained astonishing results in pattern recognition, computer vision, natural language processing, and other complex problems. Recent research shows that deep learning can be combined with reinforcement learning to solve complex problems. Deep reinforcement learning(DRL) revolutionized AI by creating an autonomous system with a higher level of understanding of the visual world. Some Real world applications are self driving cars and control policy of robots by only taking camera input. However, a major limitation of such applications is they require massive amounts of training data and are very slow to learn the task. The present objective is thus to develop a deep RL agent that can adapt rapidly to new tasks using recent reseach. To develop this agent, I created a modular framework in which various combinations of RL algorithms, different training strategies, and configurations of neural networks are trained. To test these deep reinforcement learning(DRL) systems, we generally use simulated environments and games(we can not initially test this system in the real world). In this project I will be using OpenAI Gym's environment and games to test the DRL systems. Detailed log files and results are preserved in a uniform format that permits analysis and comparison of learning performance and performance playing the video game. By this analysis, the agent is built with the best performing combination of RL algorithms, neural network configuration, and training strategies. This gives a single set of hyperparameters that will perform well in different environments and can be extended to real world applications like self driving cars and robotics.

Playing the games is a Markov Decision Process, where time is modeled by discrete intervals and the game moves from state to state partially in response to the agent and partly randomly. The agent receives state information from the "world" (the game it is trying to play). The agent produces an action (controls something in the game), then receives back a reward value (which can be negative) and the new state of the game. A key part of reinforcement learning is recurrence equations which allow rewards to be propagated backward. The agent may not get much reward until it wins or loses. If at time 10 the game is won, for example, then that reward is propagated backward to inform future choices for the intermediate states times 9, 8, 7, etc which later resulted in the win. The agent then plays another game. Sometimes it "exploits" (picks the best move in a given situation based on prior experience) and sometimes "explores"

(picks a random move, exploring new paths which may result in improving the model. Iterating, exploiting and exploring, and accumulating the backward learned rewards for all the intermediate states. The agent will learn to perform a sequence of actions in a given environment to maximize the reward.

A little more formally, at each time step agent takes an action based on the policy $\pi(a_t|s_t)$, where s_t is the current state and at the action taken. The environment reacts with the next state s_{t+1} and reward r_{t+1}. The goal is to improve policy to maximize the overall reward . To improve the policy, I have used Q-Learning. Q-Learning is based on the action-value function (or Q-function) of a policy, Q(s, a). Q-function measures the expected reward from state s by taking action a. A naive learning function would simply remember for each state the best-move-learned-so-far, along with its expected reward. The Q function remembers an expected reward from each possible action from a given state. Q is thus a "dynamic programming" algorithm. It remembers a polynomial amount of expected reward information to optimize learning over a combinatorial large number of possible paths of actions and states.

The Deep Q-Network algorithm was developed by Google DeepMind in 2015, it was able to play a wide range of Atari games by combining the deep neural network and Q-learning algorithms. To estimate the Q-values the neural network is trained. By minimizing the neural network loss Q-values are improved.

For setting up the environment, I have used OpenAI gym Library, which contains a list of game environments. This library is mainly used as a standard benchmark test for reinforcement learning algorithms. In OpenAI gym, the state of the game environment can be extracted in two forms. One form is parameterized information such as the position and velocity of a game object. The other possible way to extract the state is as a raw pixel image, the image that a human player would see and react to. I have used raw image input. To process these images, convolutional layers are added to neural networks to digest useful information out of the image.

The framework was written in Python, using PyTorch and individual loadable modules. In the framework, different configurations and parameters of the neural network, training strategies, and even different RL algorithms can be easily changed. Detailed log file of episodes (with rewards, # of time steps, loss) created. The framework allows me to use the same code for different environments. Thus the same agent configurations can play different games. Also, it's

easy to compare the performance between different configurations.

Using this framework I am able to determine the network configuration, training strategies, and reinforcement learning parameters which work well generically across multiple games. Using this results(hyperparameters) can be extended to autonomous systems and robotics which is outside scope of this project.

References:

1. Miguel Morales. (2020). Grokking Deep Reinforcement Learning book.
2. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller. (2013). Playing Atari with Deep Reinforcement Learning paper. https://arxiv.org/abs/1312.5602