Reinforcement Model for Games/Simulations

Travis James Smith

**Problem Statement and High-Level Overview:**

The purpose of this project is to explore the application of Deep Q-Learning Networks (DQN) in the training of agents to play games and perform simulated tasks. This is a well-defined task that’s common in the literature, but optimized models are still somewhat difficult to implement and train.

We’ll be implementing two models — the first is a more basic DQN, without some of the optimization techniques required to learn more difficult tasks. The goal of this model is to play a game called “Lunar Lander,” in which the agent has to guide a simulated lunar landing module to a designated landing spot. More information about the task can be found here: <https://gym.openai.com/envs/LunarLander-v2/>

For our second experiment, we’ll attempt to train an agent to play the classic Atari game “Breakout.” The model needed for this task is more intricate, for a number of reasons, and requires optimizations not present in the lunar-lander model. We’ll use a Dueling DQN with Prioritized Experience Replay. This model *might not* converge at an optimal solution due to the added mental overhead and training time, but we’ll try! For more information about “Breakout,” see here: <https://gym.openai.com/envs/Breakout-v0/>

**Hardware:** Intel Core i9-9900k @3.60GHz (16 CPUs), NVIDIA GeForce RTX 2080Ti, 32GB RAM

**Software:** Tensorflow 2.2, OpenAI Gym 0.17.2 (included as a local directory in the project)

*Note: The OpenAI Gym environment visualizations are not available out of the box on Windows and require additional setup. Gym is generally not supported on Windows at all, but training or testing a model using their Python scripts works fine. It’s strongly recommended you use macOS or Linux if you’d like to see visualizations for the agents in action.*

**References:**

*Theoretical Background:*

Richard S. Sutton and Andrew G. Barto, “[Reinforcement Learning: An Introduction.](http://incompleteideas.net/book/the-book-2nd.html)”

*Implementation Guidance:*

Lilian Weng, “[Implementing Deep Reinforcement Learning Models with…](https://lilianweng.github.io/lil-log/2018/05/05/implementing-deep-reinforcement-learning-models.html)”

Ray Heberer, “[Why Going from Implementing Q-learning to Deep Q-learning…](https://towardsdatascience.com/why-going-from-implementing-q-learning-to-deep-q-learning-can-be-difficult-36e7ea1648af)”

*Machine Learning with Phil*, “[Deep Q Learning is Simple with Keras | Tutorial.](https://www.youtube.com/watch?v=5fHngyN8Qhw&t=1411s)”

Sebastian Theiler, “[Building a Powerful DQN in TensorFlow 2.0.](https://medium.com/analytics-vidhya/building-a-powerful-dqn-in-tensorflow-2-0-explanation-tutorial-d48ea8f3177a)”

**Lessons Learned & Pros/Cons:**

Reinforcement model training is distinct from supervised learning in a variety of important ways, and the episodic nature of the process makes it quite slow. The structures and techniques used to train the function-approximating neural networks are similar to those used in supervised learning tasks, but there’s a large body of literature (centered around concepts from control theory) that informs how those deep learning techniques can be effectively applied.

For this reason, I feel that extra background is needed when approaching reinforcement learning from a deep learning perspective. See the Sutton/Barto book above for the standard introduction to the topic.

**Youtube URLs:**