Frozen Lakes

Greg Giordano
CSPB 3202 Summer 2020
https://github.com/giordang/CSPB3202_Project

Overview. The goal of this project was to implement and compare the reinforcement learning techniques of Q-learning and Deep Q Networks (DQNs) in the OpenAI Gym FrozenLake environments.

Approach. This project uses the FrozenLake-v0 and the FrozenLake8x8-v0 environments. The FrozenLake-v0 environment consists of a 4x4 grid of tiles. The agent controls a character via four possible actions on the grid with the object of reaching the goal tile without falling into the water. One point is awarded for reaching the goal tile, and no points are awarded if the character falls into the water. The FrozenLake8x8 environment consists of the same rules and objectives, only on a larger 8x8 grid. Both of these environments have a slipperiness toggle to set the character movement to be deterministic or have some level of randomness.

Given the relatively small state space of these environments, q-learning was used as a starting point. A q-table was built for each environment: 4 actions x 16 tiles for the 4x4 grid and 4 actions x 64 tiles for the 8x8 grid. An epsilon greedy policy was used to select either random or optimal actions. The epsilon was set to decay with additional training episodes.

A DQN was implemented for both the 4x4 and 8x8 slippery grids. The final model uses an embedding layer and a layer to reshape. An epsilon greedy policy with a decaying epsilon was also used in these models.

Result. Q-learning was especially effective for the non-slippery environments, able to achieve 100% completion with a relatively low number of training episodes. The slippery environments were a greater challenge, requiring more training time and only achieving ~78% completion on the 4x4 grid and 26% completion on the 8x8 grid.

The DQN for the 4x4 grid achieved a similar level of completion as Q-learning. However, the DQN for the 8x8 grid achieved a much higher level of completion (70%) compared to Q-learning. Also, the training time required for the DQN on the 8x8 grid was significantly less than Q-learning.

The starting epsilon value and the rate of epsilon decay was one of the iterative improvements of this project. Trial and error revealed that a high starting epsilon was needed with a slow decay to encourage random choice early in training and optimal choice later in training. The DQN layers were another element that required tuning. Additional dense layers with relu, softmax, and linear activation functions were attempted, but did not perform as well as the embedding layer. The number of training episodes was adjusted for each environment to try to optimize performance, however technical limitations capped the maximum number of episodes used.

Conclusion. This project shows some of the limitations of Q-learning. As the state space of a problem increases, building a q-table for all possible state, action pairs becomes inefficient. In some cases, a DQN may be used to develop an effective model in significantly less time.

This project could have been improved by using the Kaggle or Google Colab environment for training. Having access to a GPU would have reduced training time, allowing for more experimentation and increased training episodes. More experimentation with a reduced discount rate may have helped push the agent to the goal in fewer steps. These methods could also have been applied to more complex environments. The environment also has no penalty for falling into the water, only a reward for a successful completion. Penalizing falls may have improved model effectiveness.

I enjoyed working on this project, but I found adapting to the OpenAI framework challenging. I experimented with a number of environments before final settling on the FrozenLake for its simplicity. I thought it would be a good starting point for implementing a DQN in particular. I intended to apply the same methods to a Box2D or classic control problem, however this simple environment proved challenging enough to me to consume all my time.

References.

https://gym.openai.com/envs/FrozenLake-v0/
https://gym.openai.com/envs/FrozenLake8x8-v0/
https://tiewkh.github.io/blog/deepqlearning-openaitaxi/
https://towardsdatascience.com/reinforcement-learning-w-keras-openai-dqns-1eed3a5338c