Udacity Deep Reinforcement Learning P1 on Navigation
https://github.com/ckgann/DRL_P1_Navigation
Conrad Gann
August 17th, 2017

Project Details

This project uses Unity Machine Learning Agents (ML-Agents) to train a Deep Reinforcement Learning algo which chooses yellow bananas instead of blue banana.

The environment state space of size 37 and an action space of size 4 (up, down, right, left). Each episode was capped at 500 steps. Epsilon began at .7 on the first episode and decayed by 0.995 in each subsequent episode up to a minimum of 0.1.

The score is +1 for a yellow banana and -1 for a blue banana. The goal is to collect a score of +13 over 100 steps.

Source and Dependencies:
This repository was originally sourced from
https://github.com/udacity/deep-reinforcement-learning#dependencies
To run this depository I downlowded the Unity app Banana from
https://s3-us-west-1.amazonaws.com/udacity-drlnd/P1/Banana/Banana.app.zip

Relative to the original settings I changed the following parameters in Agent.py:

Agent Params Buffer = 100000
Batch Size= 100
Gamma= 0.99
Tau= 0.005
LR= 0.0055
Update Every= 5

This code was run via a jupyter notebook using the Udacity DRLND conda environment.

To train the agent run via Conrad_P1.ipynb. Conrad_P1.ipynb will call on p1_agent.py for the Unity Agent.

The Agent will train a q-table based on the weighted average of two local and target q-tables. The local and targe q-tables are weighted by the TAU and 1-TAU to form the final q-table. Each of the local and target networks has an input layer of dimension 37 equal to the state space, then two fully connected layers of width 64 which both use relu activation on the forward pass, and finally a 4 element output layer, one for each action in the q-table. The weights are trained using back propagation of the rewards returned from the Unity environment as targets. The optimization is performed using the Adam optimizer.

My agent solved the threshold test after 220 episodes with an average score of 13.34. In another run with the same settings it solved in 826 episodes with a score of 13.08.

Ideas for Future Work

This program could be further refined with a more systematic approach to parameter tuning. Without a systematic approach to parameter tuning it is difficult to evaluate any particular model.

One way to systemize the parameter tuning is create a new Hyper-DQN which has a state space equal to the hyper-parameters of the original DQN. The actions for this Hyper-DQN would be to go up or down one increment for each of the hyper parameters in the Hyper-DQN state space. The targets of this Hyper-DQN would be the scores returned by the Unity Agent for each set of hyper parameters.

Other options for future work include Experienced Replay or Dueling DQN.