

Report - Navigation (P1) Project

Udacity - Deep Reinforcement Learning Nanodegree

For: **Narendra Devta Prasanna**

For this project, we have to train an agent using DQN to pick yellow bananas and avoid purple bananas in the 'Banana' Unity-ML environment. We train the agent using three DQN approaches that all use DNN for state-action value estimation.

The state observation is in the form of a 37-D state vector. The state action space consists of 4 discrete actions: walk forward, walk backward, turn left and turn right. More details are in the README.md file provided as part of the project description.

Implementation:

The first implementation uses the simplest DQN approach [Minh 15] as outlined in the Nature-2015 paper. It utilizes an experience replay buffer and distinct DNN networks for Q-target and Q-local. The code for this implementation and the hyperparameters are directly re-used from the course assignment that was used for solving OpenAI's Lunar-lander environment. The DNN architecture is also adapted directly from the same course project and consists of a 3-layer MLP (input layer + 2 hidden layer + output layer).

Python notebook	Navigation_dqn.ipynb
Model	model_dqn.py
Agent	dqn_agent.py
Trained network weights	

The second implementation is based on the Double DQN (DDQN) approach [Hasselt 15]. Function approximation techniques, especially those that utilize bootstrapping such as the TD(0) based approach used here, tend to suffer from overestimation of the state-action value function which can result in sub-optimal learning. To alleviate this, when the target Q-value is calculated, two different networks are used to first determine the optimal action and then to determine the action value. In this case, the action to be taken is determined from the local network and the Q-value of that action is derived from the target network. The DNN architecture is the same MLP as that used for DQN implementation above.

Python notebook	Navigation_ddqn.ipynb
Model	model_ddqn.py
Agent	ddqn_agent.py
Trained network weights	<u>model_ddqn.pt</u>

The third and final implementation is based on the dueling network architecture [Wang 16] and also combined with the DDQN approach. The dueling network architecture explicitly separates the representation of the state value and the action advantage (state-dependent) value while sharing a common feature learning network. The separation between state value and action advantage representation leads to more frequent updates to the value function and thus faster learning and also more robust learning resulting in a more optimal policy. The DNN architecture has a common feature trunk consisting of two hidden layers. The output of the feature estimator feeds two parallel branches consisting of one hidden layer and an output layer each for state value estimate and action-advantage value estimate.

Python notebook	Navigation.ipynb / Navigation_dueling_ddqn.ipynb
Model	model.py / model_dueling_ddqn.py
Agent	dueling_ddqn_agent.py
Trained network weights	<u>model.pt</u> / <u>model_dueling_ddqn.pt</u>

Training and Results:

First, each of the three implementations is let to run for 2000 episodes and the maximum agent score is observed as well as the learning. The score observed is the moving average over the past 100 episodes. From these trial runs it is observed that both DQN and DDQN implementations achieve a maximum score (moving average of 100 episodes) of between 16 and 17. However, the score oscillates more in the case of DQN than for DDQN implementation. For the case of Dueling+DDQN implementation, the max score observed is between 17 and 18 which is higher than what is achievable for either DQN or DDQN approaches.

Then the agent is trained for a second time till it reaches the target score. The target score for DQN and DDQN is set to 16 and it is set to 17 for Dueling+DDQN implementation. The results are summarized in the table below. Training is done on a Mac-OS (High Sierra) machine with Nvidia GTX 780M GPU and Pytorch version 1.1. GPU is enabled for the training.

Model	Run time (seconds)	Num. Episodes	Agent Score (moving avg of 100 episodes)
DQN	970	970	16
DDQN	875	860	16
Dueling+DDQN	1065	898	17

References:

[Minh 15]: Human-level control through deep reinforcement learning, Vol 518, Nature, Feb 2015
[Hasselt 15]: Deep Reinforcement Learning with Double Q-learning, Dec 2015, Arxiv
[Wang 16]: Dueling Network Architectures for Deep Reinforcement Learning, Apr 2016, Arxiv