Udacity Navigation Project

Thomas Lecat

April 2021

1 Environment

The environment is described as Markov Decision Process (S, A, R, p, γ) with unknown dynamics. The goal is to approximate the optimal policy π^* that maximizes the upcoming cumulative reward in every state.
This project uses value based methods, which seek to approximate the optimal Q value Q^*, defined as:

$$Q^*(s,a) = \max_\pi \mathbb{E}_\pi(\sum_t \gamma^t r_t | S_0 = s, A_0 = a), \forall (a,s) \in A \times S$$

Given Q^*, one can derive π^* to solve the problem: $\pi^*(s) = \arg\max_a Q^*(s,a), \forall s \in S$

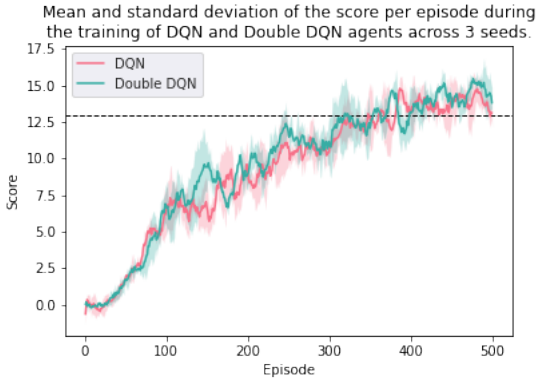
2 Agent

2.1 DQN

A vanilla DQN agent is implemented as the base solution.
DQN approximates the Q value by a neural network parameterized by θ. Its loss function is defined as the mean squared TD error of the Bellman optimality equation:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}\left[((r + \gamma \max_{a'} Q(s', a'|\theta^-)) - Q(s,a|\theta))^2\right]$$

where θ^- are the parameters of a target network, slowly updated towards θ at every training step. This target network makes the TD target $r+\gamma \max_{a'} Q(s', a'|\theta^-)$ independent from the parameters θ, which stabilizes the training.
This loss is minimized by an Adam optimizer, a variant of the Stochastic Gradient Descent algorithm. At each step, the gradient is approximated from a minibatch of experiences $(s, a, r, s', d)_i$. To avoid a strong correlation between the steps of the minibatch, transitions are stored in a large replay buffer during rollouts and then sampled uniformly.



2.2 Double DQN

The DQN agent is implemented in a modular way, to enable extension from recent papers to be added later.
As a start, we implemented the improvements from the Double DQN paper: Instead of using the target network to chose the next action and estimate its value in the target, use Q_θ to chose the action, and Q_{θ^-} to estimate its value.

3 Hyperparemeters

The network architecture is composed of 2 fully connected layers with 64 neurons each. The agent follows an ϵ-greedy policy, with epsilon gradually decreasing from 1 to 0.1 over $30,000$ steps.
The table below summarizes the main hyperparameters:

Hyperparameters			
Parameter		Vanilla DQN	Double DQN
learning rate	α	5e-04	
discount factor	γ	0.99	
target network update coefficient	τ	0.001	
buffer size		100,000	
batch size		64	

4 Results

Both vanilla DQN and double DQN have been trained with 3 seeds. The reward curves are averaged over seeds and plotted below:
The addition of the double DQN strategy doesn't improve the results much and both agents manage to solve the environment in less than 400 episodes.

5 Next steps

The next steps for the project consist in implemented other DQN extensions such as Dueling DQN and Prioritized Experience Replay. Hyperparemeters could also be further tuned using strategies such as HyperOpt or Population Based Training.