COMP 412 - Autonomous Agents - Term Project

Landing Spaceships with Q-Learning

Kariotakis Emmanouil

Instructor: Michail G. Lagoudakis

School of Electrical and Computer Engineering
Technical University of Crete

27 February, 2022

Overview

1. Theoretical Background
   - Q-Learning
   - Deep Q-Learning
   - Dueling Deep Q-Network
2. Environment
   - LunarLander-v2
3. Implementation
4. Results
5. Conclusion
   - Advantages - Disadvantages
   - Challenges

Q-Learning

Variables Definition

- s_t: state in iteration t
- a_t: action in iteration t
- r_t: reward in iteration t
- γ: discount factor $(0 < \gamma \leq 1)$
- α: learning rate

Watkins et al. 1989 [1]

- for each (s_t, a_t, r, s_{t+1}) sample:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_t + \gamma \max_{a_{t+1} \in A} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right),$$

- if s_{t+1} is a terminal state:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_{t+1} - Q(s_t, a_t) \right).$$

Deep Q-Learning

The basic idea behind DeepMind's algorithms [2], [3].

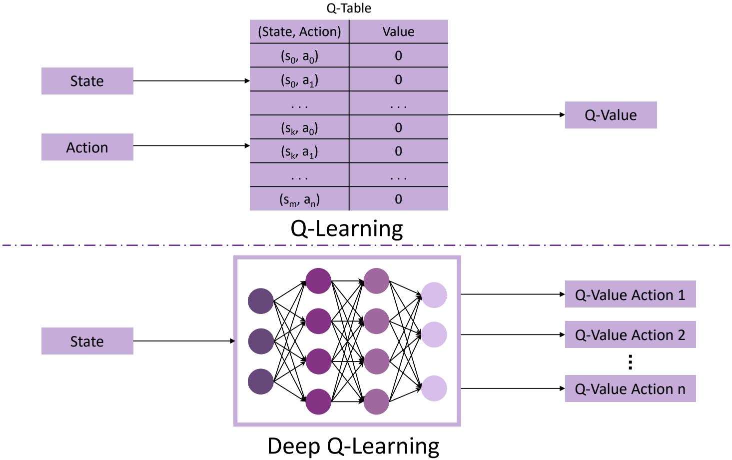


Figure: Q-Learning & Deep Q-Learning

Deep Q-Learning

Creating a more stable model:

- Target Network [4]: a copy of the estimated value function that is held fixed to serve as a stable target for some number of steps.
- Experience Replay: instead of training the agent after each episode, we store its experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ in a table, called Replay Buffer.
- Epsilon-Greedy Action Selection: determining the amount of exploration vs exploitation.

Dueling Deep Q-Network

Google DeepMind, 2016 [5]

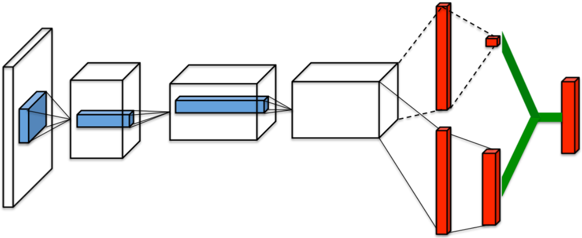


Figure: Dueling Q-Network

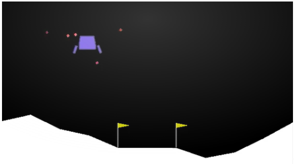
Connection of the two streams to the output of the network ($V(s)$ first stream, $A(s,a)$ second stream):

$$Q(s,a) = V(s) + \left(A(s,a) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s,a') \right),$$

where $|\mathcal{A}|$ is the number of available actions.

LunarLander-v2

The environment is provided by OpenAI Gym toolkit [6].
The **goal** of the task is to land a spaceship between the two yellow flags without crashing it.



Actions

- fire left orientation engine
- fire main engine
- fire right orientation engine
- do nothing

States

- x-axis coordinates
- y-axis coordinates
- x-axis linear velocity
- y-axis linear velocity
- angle
- angular velocity
- left leg in contact with the ground (boolean)
- right leg in contact with the ground (boolean)

LunarLander-v2

Rewards

- Landing between flags and coming to rest: +100-140 points.
- Crashing: -100 points.
- Coming to rest: $+100$ points.
- Each leg with ground contact: $+10$ points.
- Firing the main engine: -0.3 points each frame.
- Firing the side engine: -0.03 points each frame.
- Solved: 200 points.

Termination if the lander

- crashes (its body gets in contact with the moon),
- gets outside of the viewpoint (x-coordinate is greater than 1),
- is not awake (doesn't move and doesn't collide with any other body).

Implementation

- In Python 3.9.10, using PyTorch 1.10.2.
- Both DQN and Dueling-DQN.

Hyperparameter	**Value**
Replay Buffer Size	10^5
Batch Size	64
Discount Factor γ	0.99
Update Factor τ	10^{-3}
Learning Rate α	$5 \cdot 10^{-4}$
ϵ-start	1
ϵ-end	0.01
ϵ-decay	0.995

Table: Hyperparameters

Results

- **Score**: sum of all rewards from each step in each episode.
- Problem solved at about, score=250.

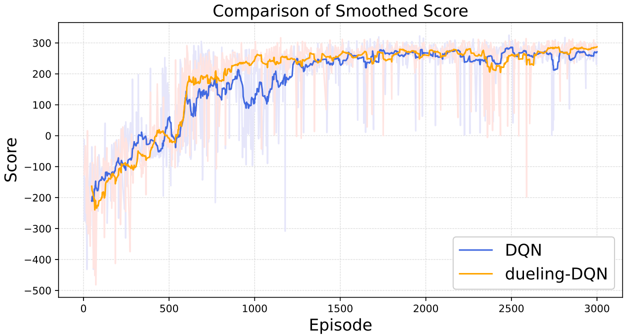


Figure: DQN vs Dueling-DQN

- Dueling-DQN gets faster to a score 250 than simple DQN.
- Clearer difference when we increase the number of actions [5].

Conclusion

Advantages

- Does not need to any knowledge about the environment.
- It can be used for solving plenty different problems with changing only the number of states and actions, based on the given environment.

Disadvantages

- Needs many episodes to be trained properly (it may even take days of training for large state and action spaces).
- Unable to solve problems that require a continuous action space (only with discretization).

Challenges

- Tweaking of hyperparameters.
- Creating an environment nad engineering its observations and rewards for the agent

References

[1] C. Watkins, "Learning from delayed rewards,", Jan. 1989.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. DOI: 10.1038/nature14236. [Online]. Available: https://doi.org/10.1038/nature14236.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. arXiv: 1312.5602. [Online]. Available: http://arxiv.org/abs/1312.5602.

[4] S. Kim, K. Asadi, M. Littman, and G. Konidaris, "DeepMellow: Removing the need for a target network in deep q-learning," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, International Joint Conferences on Artificial Intelligence Organization, Aug. 2019. DOI: 10.24963/ijcai.2019/379. [Online]. Available: https://doi.org/10.24963/ijcai.2019/379.

[5] Z. Wang, N. de Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," *CoRR*, vol. abs/1511.06581, 2015. arXiv: 1511.06581. [Online]. Available: http://arxiv.org/abs/1511.06581.

[6] G. Brockman, V. Cheung, L. Pettersson, *et al.*, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016. arXiv: 1606.01540. [Online]. Available: http://arxiv.org/abs/1606.01540.