Assignment: Deep Q Learning
Course: Reinforcement Learning, Leiden University

1 Introduction

In the previous assignment, you have worked with tabular reinforcement learning algorithms (such as q-learning, SARSA, etc.) on a grid world task. But if the state space is high-dimensional (e.g. images) or very large, storing them into a table is infeasible. In this assignment, we will look at the use of deep neural networks for *function approximation*. Learning with function approximation allows 1) for a compact representation of the solution (usually in the form of a policy or value function, which can be stored in memory in approximate form) and 2) generalization, where similar states will share information automatically. Moreover, function approximation is also a practical solution for tasks with continuous action spaces, such as robot joints.

The environment that we will use in this assignment is Cartpole as shown in Figure 1. In this environment, there is a cart (black box) that moves along a horizontal axis. The goal is to move the cart in such a way that the pole stays upright. Note that you do not have to implement this environment, as OpenAI provides it within Gym (see source below figure, or click on highlighted Cartpole text).

Figure 1: Illustration of the Cartpole environment. Source: [1].

You will implement the deep Q learning algorithm (see below) that attempts to learn a Q-value for every state-action pair $Q(s, a)$ that models the expected total outcome after performing action a in state s.

2 Deep Q-learning

Deep Q-Network (DQN) will receive as input a state s and outputs for every action a the Q-value $Q(s, a)$. This way, the network simply acts as a regular table: you "look up" the Q-value using the state as key (input) and observe the produced values (outputs) for every action. Generally, DQN is trained using experience replay and has a target network.

Implement the deep Q learning agent using whatever you are comfortable with (e.g. tensorflow, keras, pytorch), then perform a scientific ablation study. Make sure that we are able to reproduce your results via one single command, and different parts of the model could be easily turned on/off. (e.g. "`sh run_dqn.sh`" or "`python dqn.py`" or "`python dqn.py --experience_replay --target_network`" on a university machine). Think about the architecture of your network: Do you need convolutions? How many layers? How many nodes per hidden layer? You may want to experiment with various architectures and tune hyper-parameters for exploration, and report on these results to explain your final choice.

2.1 Exploration Strategy

Exploration plays a key role in reinforcement learning. Commonly used strategies are ϵ-greedy, Boltzmann exploration, annealing ϵ-greedy, etc. Implement at least two different exploration strategies for training your agent and compare them. You can reuse ϵ-greedy or Boltzmann from A1. As bonus, there are also some 'smarter' exploration strategies such as 'novelty-based' exploration and 'curiosity-based' exploration.

2.2 Experience Replay

By maintaining a replay buffer, we are able to reuse collected data multiple times. Meanwhile, sampling batches randomly from the buffer can break the correlation between consecutive data, which can make the training more stable. Implement the replay buffer for experience replay.

2.3 Target Network

During the update, we try to push the q values towards the target q values which is the immediate reward plus the bootstrapped q values. Since a single update will affect the whole network, and it will cause the target q values to become non-stationary. In order to make stationary target q values, we could use another network for providing target q values and update it using the weights of the original network periodically. Implement the target network.

2.4 Checklist

Overall, you might want to check what you should do/implement/report for this assignment:

- DQN agent with:
  - Different exploration strategies (at least two)
  - Experience replay (train with a replay buffer)
  - Target network (use another network to provide the update target)
- Tune hyper-parameters:
  - network architecture (number of layers, number of neurons)
  - learning rate
  - exploration factor
  - The three above are just examples. Think about which are relevant, reason in your report why you choose them and what their effect on learning is.
- Ablation Study: compare different models in terms of learning speed, performance, stability($-$ here means part of the model is removed, either experience replay(ER) or target network(TN) or both.):
  - Compare DQN with DQN$-$ER
  - Compare DQN with DQN$-$TN
  - Compare DQN with DQN$-$EP$-$TN

Make sure that other parameters are fixed when you perform the ablation study.

3 Bonus(optional)

The instructions on the previous section are enough to pass the assignment with good grade. Should you however want to impress us, then you are welcome to show additional experiments.

- Maybe you find other improvements of DQN and want to try them, such as double DQN or dueling DQN, etc.
- Maybe you try out new exploration strategies besides ϵ-greedy and Boltzmann
- Maybe you want to apply DQN to other environments
- Maybe you have other ideas that you would like to try. Feel free to add your own thoughts and experiments.

4 Submission

Make sure to nicely document everything that you do. Your final submission consists of:

- Source code with instructions (e.g., README) that allows us to **easily** (single command per experiment / sub task) rerun your experiments on a university machine booted into Linux (DSLab or computer lab).
- You have to use the ICML 2021 Latex template for your Report. (Obtainable here)
- A self-contained scientific pdf report of at most 8 pages with figures etc. This report contains an explanation of the techniques, your experimental design, results (performance statistics, other measurements,...), and overall conclusions, in which you briefly summarize the goal of your experiments, what you have done, and what you have observed/learned.

If you have any questions about this assignment, please visit our lab sessions on Friday where we can help you out. In case you cannot make it, you can post questions about the contents of the course on the Brightspace discussion forums, where other students can also read and reply to your questions. Personal questions (not about the content of the course!) can be sent to our email address: rl@liacs.leidenuniv.nl.

The deadline for this assignment is 02.04.2023 at 23:59. Unforeseen changes to this deadline would be announced via Brightspace. For each full 24 hours late, one full point will be deducted (e.g., if your work is graded with a 7, but you are two days too late, you get a 5).

Good luck and have fun! :-)

5 Useful resources

For this assignment you may find the following resources useful:

- OpenAI's Gym website is here. It provides a wealth of reinforcement learning examples. Browse to example Environments, and through algorithms.
- Original Deep Q learning paper from DeepMind.here
- Exploration strategies blog. here
- You may draw inspiration (not copy, that is fraud!) for your implementation from baselines.

References

[1] *OpenAI Gym Cartpole-v1*. Accessed: 27-11-2020. URL: `https://gym.openai.com/envs/CartPole-v1/`.