Deep Learning Lab Course
Forth Exercise: Reinforcement Learning

Maria Hügle

December 4, 2018

The goal of this exercise is to experiment with reinforcement learning. You will implement a DQN agent and evaluate its performance on a classic control task and the CarRacing environment from OpenAI Gym. You will get in touch with experience replay and target networks. Further, you will implement ϵ-greedy exploration for online training. Read more about Deep Q-Networks in *Playing Atari with Deep Reinforcement Learning* https://arxiv.org/pdf/1312.5602.pdf.

Please start early with this exercise and work in groups of two people. The training with Q-learning can take quite a long time for the CarRacing environment. You can run the code either on GPU or CPU. A GPU machine will be faster when you work with images and CNNs, but you should be able to get good results in around 6-8 hours of compute on a CPU.

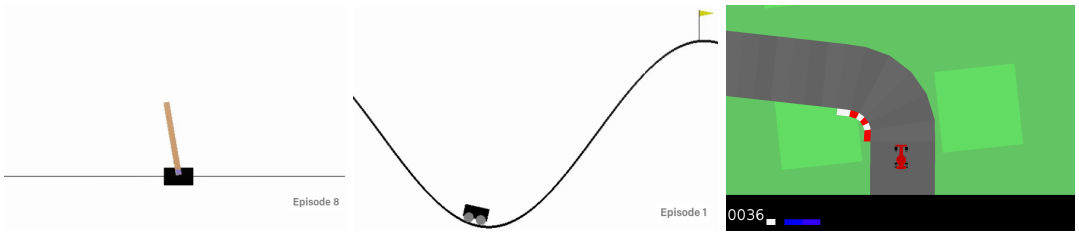

Figure 1: OpenAI Classic Control Environments and CarRacing

At the end, send us a link to your github repository, containing your code and a report (1 or 2 pages) with your experiments and results. Submit this exercise until the 08.01.2019 via mail to hueglem@cs.uni-freiburg.de and eitel@cs.uni-freiburg.de.

1 Getting Set Up

The code can be found at https://github.com/aisrobots/dl-lab-2018. We use the same set up as in exercise sheet 3. You will work with TensorFlow, OpenAI Gym with Box2d and Tensorboard.

2 Reinforcement Learning: Deep Q-Networks

2.1 CartPole

We start with CartPole, a classic control task from gym. Implement DQN by extending the starter code in our repository. We provided you some components of Q-Learning. Use `train_cartpole.py` for the training and finish the DQN agent in `dqn/dqn_agent.py`. For CartPole, you will find a network in `networks.py` and code for the replay buffer `dqn/replay_buffer.py`

You can watch the progress of the total episode reward (achieved with ϵ-greedy exploration) during the training in tensorboard. Additionally, evaluate the episode reward every N episodes with greedy actions only (compute the mean episode reward of the agent with deterministic actions over 5 episodes). Show both tensorboard plots in your report.

Optional: Train DQN also on one other gym environment with discrete actions, e.g. MountainCar. See https://gym.openai.com/envs/#classic_control.

2.2 CarRacing

After DQN works for CartPole, you can start working on the CarRacing environment. Implement a 2-3 layer CNN with strided, no padding convolutions and one fully-connected layer for your Q-network and target network. You'll find starter code for the training in `train_carracing.py`. Add a maximum capacity (with first-in-first-out) to the replay buffer to avoid memory errors if you collect a lot of data.

Getting good results with DQN for the CarRacing environment is harder than for CartPole. Only changing the network architecture to CNNs probably won't lead to good results for the RacingCar. However, start with exactly this and watch the episode reward during the training in tensorboard for some time.

You may want to use some of the following tricks to make it work:

1. Exploration: If you turn on the rendering during training, you will see that sampling actions from a uniform distribution doesn't let the agent explore the environment properly. Use higher probabilities for accelerating or going straight.
2. Frame Skipping: skipping n frames and repeating the RL agents action during this time can help (already implemented, you only need to change `skip_frames=0` in `train_carracing.py`).
3. To speed up training, train on shorter episodes in the beginning (by adapting `max_timesteps`).

As before in exercise 2.1, show the training and evaluation episode rewards of your agent in the report. Additionally, evaluate the agent with `test_carracing.py` and show the final performance over 15 test episodes.

3 Bonus: Double Q-Learning

Besides using a target network, you can use double Q-Learning (see Deep Reinforcement Learning with Double Q-learning) to reduce the overestimation bias. Implement double Q-Learning and show the results for CartPole and CarRacing in your report.

4 Bonus: Exploration

Implement ϵ-annealing or Boltzmann exploration and compare these more sophisticated exploration methods to the ϵ-greedy exploration you used above. Show the results for CartPole and CarRacing in your report (including the training curves).

Hint: for Boltzmann exploration you sample your actions according to the distribution of your Q-values.