Reinforcement Learning
Exercise 4

October 4, 2021

1 Function approximation

In many real world scenarios, the dimensionality of the state space may be too high to compute and store the Q-values for each possible state and action in a Q-table. In addition, as you'll find out in this exercise, tabular methods may suffer from low sample efficiency.

The state value and action value functions can be learned by using a function approximator, such as a radial basis functions (RBFs) or a neural network. In this exercise, you will start with basic features and build your way up to a simple DQN.

Please start working on this assignment early and don't leave it for the last moment, as the DQN part will take some time to train.

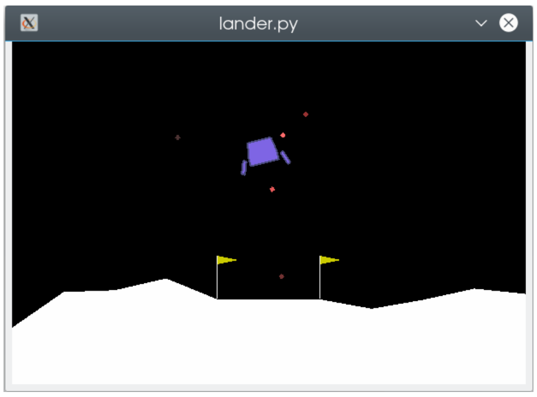

Figure 1: The Lunar lander environment.

Aalto University
School of Electrical Engineering

Reinforcement Learning course staff
Intelligent Robotics Group
aalto.fi, irobotics.aalto.fi

1.1 Radial Basis Functions

A radial basis function (RBF) is a real-valued function ϕ that maps a vector $\boldsymbol{x} \in \mathbb{R}^d$ into a real-value $\phi : \mathbb{R}^d \mapsto \mathbb{R}$. The RBF $\phi(\boldsymbol{x}) = \phi(||\boldsymbol{x}||)$ acts on a distance (radial) and a basis function ϕ (e.g. Gaussian function). The RBFs are commonly used as kernels in machine learning. The RBF kernel is defined as a $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$, where ϕ is the Gaussian function. Therefore, the RBF kernel is defined as

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\frac{-||\boldsymbol{x} - \boldsymbol{x}'||^2}{2\sigma^2}), \tag{1}$$

where the parameter σ^2 is the variance and gives shape to the Gaussian function.

On the exercise you are given a `featurizer` which uses an `RBFSampler`. The `RBFSampler` uses a Monte Carlo approximation of the RBF kernel and generates samples with the specified variance. The `RBFSampler` has been trained on a predefined dataset. Thus, the RBF kernel will map the state $\boldsymbol{x} \in \mathbb{R}^d$ to the distance to the trained dataset. Therefore, an input state will be transformed to as many features as samples N are specified in the `RBFSampler` $\phi : \mathbb{R}^d \mapsto \mathbb{R}^N$.

2 Approximation with non-linear features

Task 1 - 15 points Implement Q-learning using function approximation, at every timestep using the latest state transition to perform a TD(0) update. Also implement ϵ-greedy action selection. Test the implementation on the Cartpole environment. Test two different features for state representations:

(a) handcrafted feature vector $\phi(s) = [s, |s|]^T$,
(b) radial basis function representations (use the `featurizer` inside the Agent class).

Hint: You might need to install *scikit-learn*. It can be done by running: `pip install -U scikit-learn`.

Hint: The training plot of the agent is initially displayed using Tensorboard, the results can be shown running: '`tensorboard -logdir runs`' and entering into the website address prompted, e.g. `http://localhost:6006/`

Question 1 - 10 points Would it be possible to learn **accurately** Q-values for the Cartpole problem using linear features (by passing the state directly to a linear regressor)? **Why/why not?**

Task 2 - 15 points Modify your Task 1 implementation to perform minibatch updates and use experience replay (**while keeping the original code for Task 1 submission**) [1, p. 440]. Run the experiments with Cartpole with both feature representations.

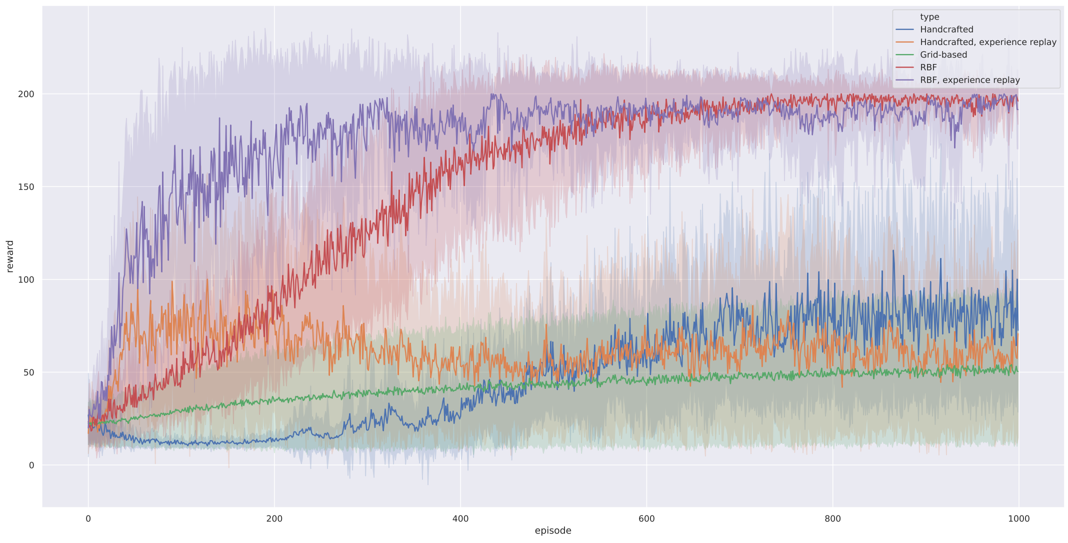

Figure 2: Training process with different methods

Hint: Pass the elements to the `store_transition` function in the following order: s, a, s', r, $done$.

Question 2 Figure 2 shows the training performance of all four methods from Task 1, together with grid-based learning from Exercise 3 evaluated using GLIE with $a = 50$.

Question 2.1 - 10 points How does the experience replay affect the learning performance?

Question 2.2 - 5 points Discuss the benefits and cons of using hand-crafted features. As an example, you can refer to the given hand-crafted feature and more complex features like $\phi(s) = [s_x, s_{\dot{x}}, \cos(s_\theta), \sin(s_\theta), s_{\dot{\theta}}]^T$

Question 2.3 - 10 points Do grid based methods look sample-efficient compared to any of the function approximation methods? Why/why not?

Hint: An algorithm is said to be sample-efficient when it requires less samples (data) to reach an optimal performance.

Task 3 - 10 points Create a 2D plot of policy (best action in terms of state) learned with RBF with experience replay in terms of x and θ for $\dot{x} = 0$ and $\dot{\theta} = 0$.

Hint: You can discretize the state-space for plotting for $\dot{x} = 0$ and $\dot{\theta} = 0$.

3 A (not-so-)deep Q-network

Task 4 - 5 points Replace the RBFs in your Task 2 implementation with a neural network (**while keeping the original code for Task 2 submission**). A basic DQN implementation can be found in `dqn_agent.py`. Evaluate the method's performance in CartPole and LunarLander environments.

Question 3.1 - 5 points Can Q-learning be used directly in environments with continuous action spaces?

Question 3.2 - 15 points Which steps of the algorithm would be difficult to compute in case of a continuous action space? If any, what could be done to solve them?

Hint: You can go through the DQN algorithm details in the paper [2], it might help you also with the project!

4 Submission

The deadline to submit the solutions through MyCourses is on Monday, 25.10 at 12:00.

The report *must* include:

1. **Answers to all questions** posed in the text (remember to answer all questions, specially in cases where there is a why/why not).
2. **Training plots** of all methods (Task 1 a), b), Task 2 and Task 4).
3. **Policy plot** from Task 3.

In addition to the report, you must submit as separate files, in the same folder as the report:

1. Python code used to solve **all task exercises**.

Please remember that not submitting a PDF report following the **Latex template** provided by us will lead to subtraction of points.

For more formatting guidelines and general tips please refer to the submission instructions file on mycourses.

If you need help or clarification solving the exercises, you are welcome to come to the exercise sessions in weeks 40/41/42. Good luck!

References

[1] Sutton, Richard S., and Andrew G. Barto. "Reinforcement Learning: An Introduction (in progress)." London, England (2017). `http://incompleteideas.net/book/RLbook2018.pdf`

[2] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013). `https://arxiv.org/pdf/1312.5602.pdf`

5 FAQ

How should T4 training look for CartPole and LunarLander? See Figure 3, and Figure 4.

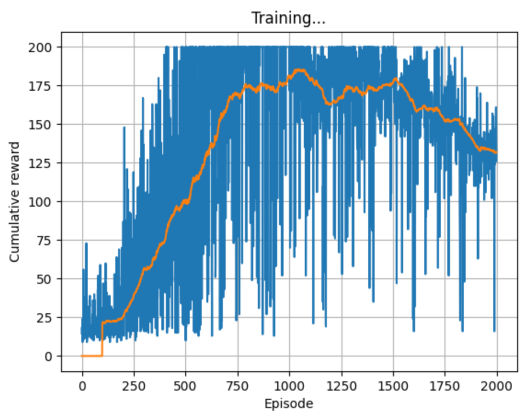

Figure 3: DQN CartPole training results for sample test.

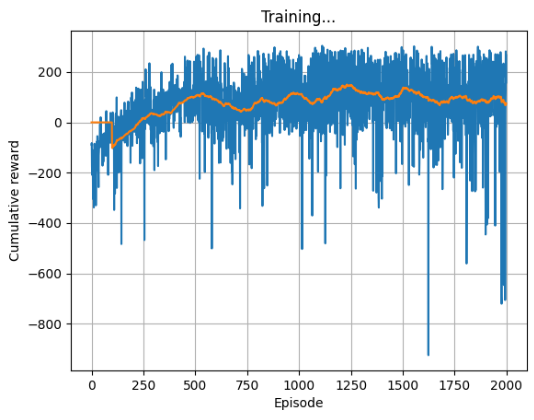

Figure 4: DQN LunarLander training results for sample test.

Is it normal that the DQN Lunar Lander training takes a lot of time to train? Yes. This is due to the complexity of the network and hyper-parameters of the algorithm. This is a good time to train using Aalto's scientific resources if you need more computational resources!