Reinforcement Learning
Exercise 4

October 5, 2022

Introduction

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. Instead, 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 handcrafted features and build your way up to a simple Deep Q-Network (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.

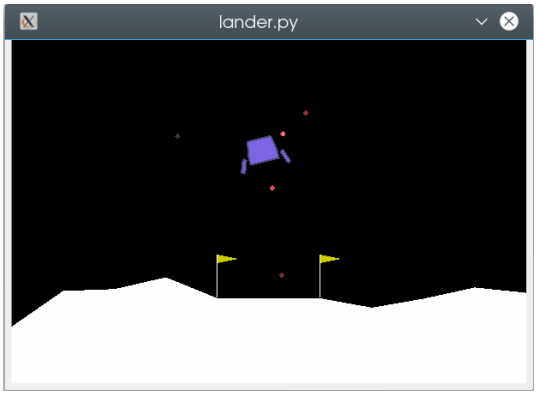

Figure 1: The Lunar lander environment.

Aalto University
School of Electrical
Engineering

Reinforcement Learning course staff
Aalto Robot Learning Lab
aalto.fi, rl.aalto.fi

Approximate with non-linear features

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` $\phi : \mathbb{R}^d \mapsto \mathbb{R}^N$.

In a nutshell, the `featurizer` function maps low-dimensional states to a high-dimensional representation, which reveals properties that haven't been shown in the original low-dimensional space, e.g., for the Cartpole environment, it maps the original four-dimensional states to a 230-dimensional vector.

Task 1 - 20 points Implement Q-learning using function approximation. Also implement ϵ-greedy action selection. 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 RBFAgent class).

For this Task you need to modify functions `featurize`, `get_action`, and `update` in `rbf_agent.py`. Test the implementation on the Cartpole environment by running the `train.py` script with default configs. You can test the trained model with the `test.py` script (set `save_video=true` to record videos during testing). **Attach the training performance plots for both (a) and (b) in your report.** See figure 2 for an example training performance plot for (b).

Question 1.1 - 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?**

Question 1.2 - 10 points In Task 1, we collect observed states, actions, and rewards into an experience replay buffer. During training we **randomly** sample mini-batches from this buffer.

Figure 2: The training performance plot for Task 1 (b) might look something like this.

Why do we use the experience replay buffer, how does it affect the learning performance? Why do we sample the mini-batches randomly?

Hint: In machine learning, when training a model, we usually assume dataset includes i.i.d samples from a unknown distribution.

Question 1.3 - 10 points In Exercise 3, we used grid-based Q-learning to balance the Cartpole. Are grid-based methods sample-efficient compared to the RBF 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 2 - 10 points Create a 2D plot of policy (best action in terms of state) learned with RBF in terms of x and θ for $\dot{x} = 0$ and $\dot{\theta} = 0$, such that θ is on the horizontal axis and x is on the vertical axis. **Attach the plot into your report.**

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

A (not-so-)deep Q-network

Task 3 - 10 points Finish the incomplete code in `dqn_agent.py` (functions `update` and `get_action`, marked with `TODO`) to implement a DQN agent. Run the `train.py` script with parameters `env_agent=cartpole_dqn` and `env_agent=lunarlander_dqn` to evaluate the DQN's performance in CartPole and LunarLander environments. **Attach the training performance plots for both environments into your report.** See figure 3 for an example training performance plot for `cartpole_dqn`.

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

Figure 3: The training performance plot for `cartpole-dqn` in Task 3 might look something like this.

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?

Question 3.3 - 10 points In DQN, we use an additional target network to calculate the target Q value. **Why** we need this? Can we just use the same network in calculating both $Q(s,a)$ and $\max_a(Q(s',\cdot))$? Also, what will happen if we do not stop gradient of the target Q value?

Submission

The deadline to submit the solutions through MyCourses is on Monday, Oct. 10 at 23:55.

Your submission should consist of (1) a **PDF report** containing **answers to the Questions** asked in these instructions and plots/model files/reported metrics **as required in each of the Tasks**, (2) **the code** with solutions used for the exercise. Please remember that submitting a PDF report without following the **Latex template** provided by us will lead to subtraction of points.

For more formatting guidelines and general tips please refer to the README.md.

If you need help or clarification solving the exercises, you are welcome to join the Slack channel and exercise sessions. If you need help or clarification solving the exercises, you are welcome to come to the exercise sessions. 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`