Exercise 4

Jari Mattila - 35260T
ELEC-E8125 - Reinforcement Learning

October 30, 2021

Task 1

The training performance plots are presented below for Task 1a using handcrafted feature vector $\phi(s) = [s, |s|]^T$ in Figure 1 and Task 1b using radial basis function representations in Figure 2.

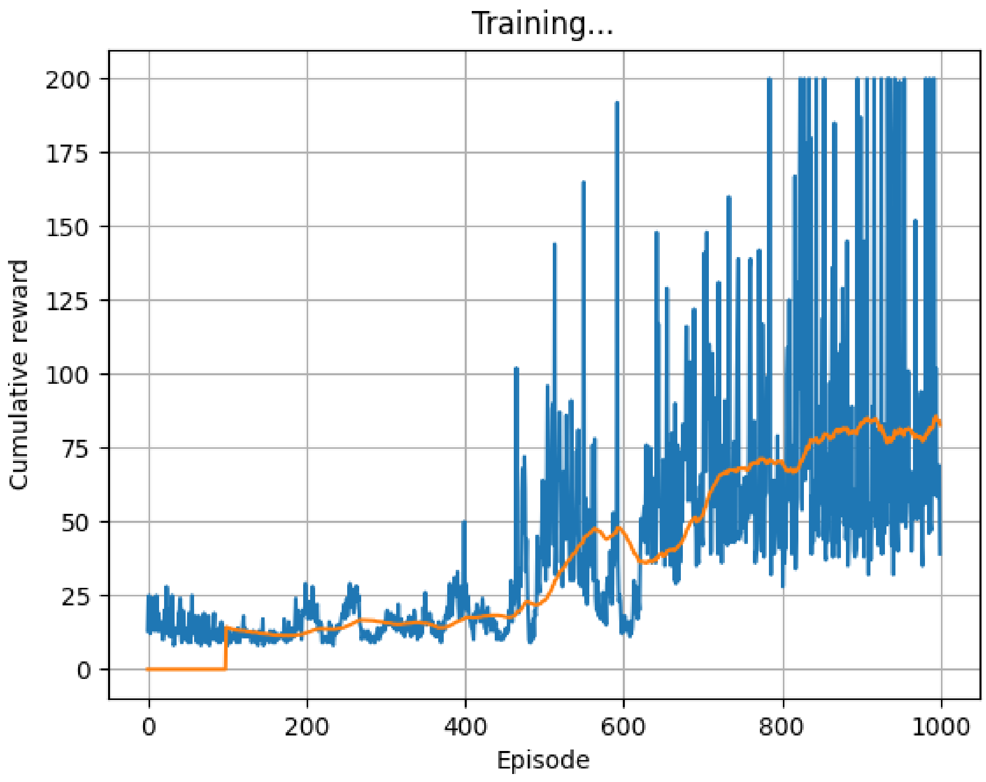


Figure 1: Training performance using handcrafted feature vector $\phi(s) = [s, |s|]^T$.

Source files: main_task1.py, rbf_agent_task1a.py, rbf_agent_task1b.py

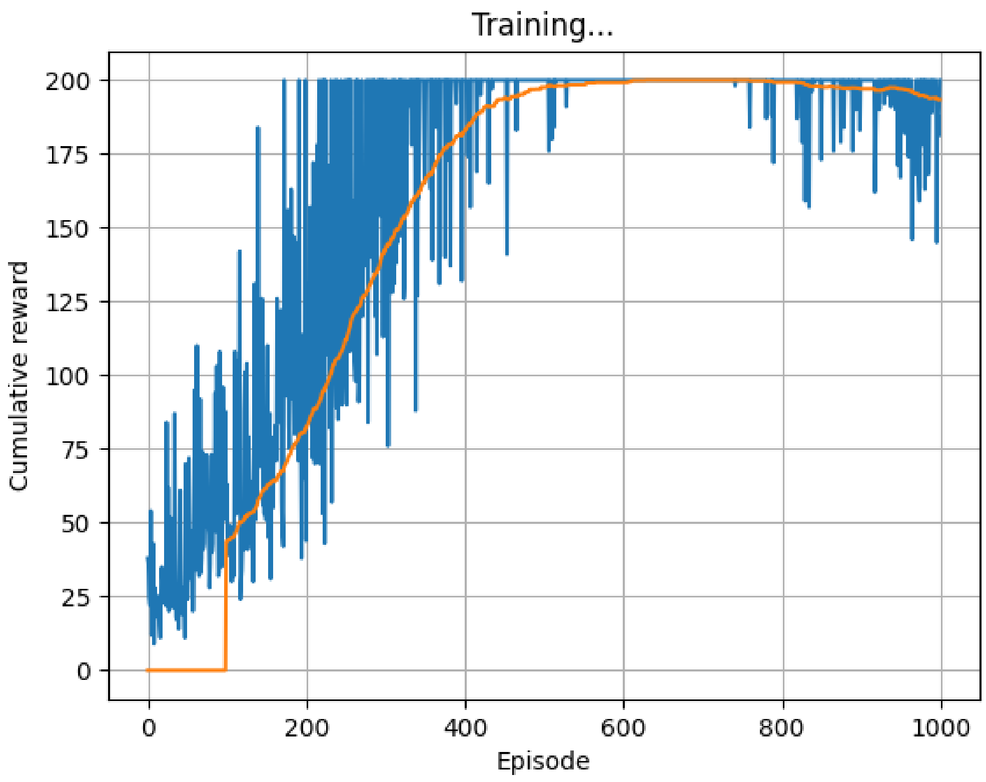


Figure 2: Training performance using radial basis function representations.

Question 1

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?

Learning Q-values for the Cartpole problem accurately using linear features can be difficult because the linear model cannot model dependencies between the features. For example, in the Cartpole problem the high angular velocity can be either good or bad depending on the angle - this cannot be taken into account by the linear model (for details see Section 9.5 of Sutton's book).

Task 2

The implementation of Task1 has been modified to perform minibatch updates and use experience replay in the following. The training performance plots are presented below for Task 2a using handcrafted feature vector $\phi(s) = [s, |s|]^T$ in Figure 3 and Task 2b using radial basis function representations in Figure 4.

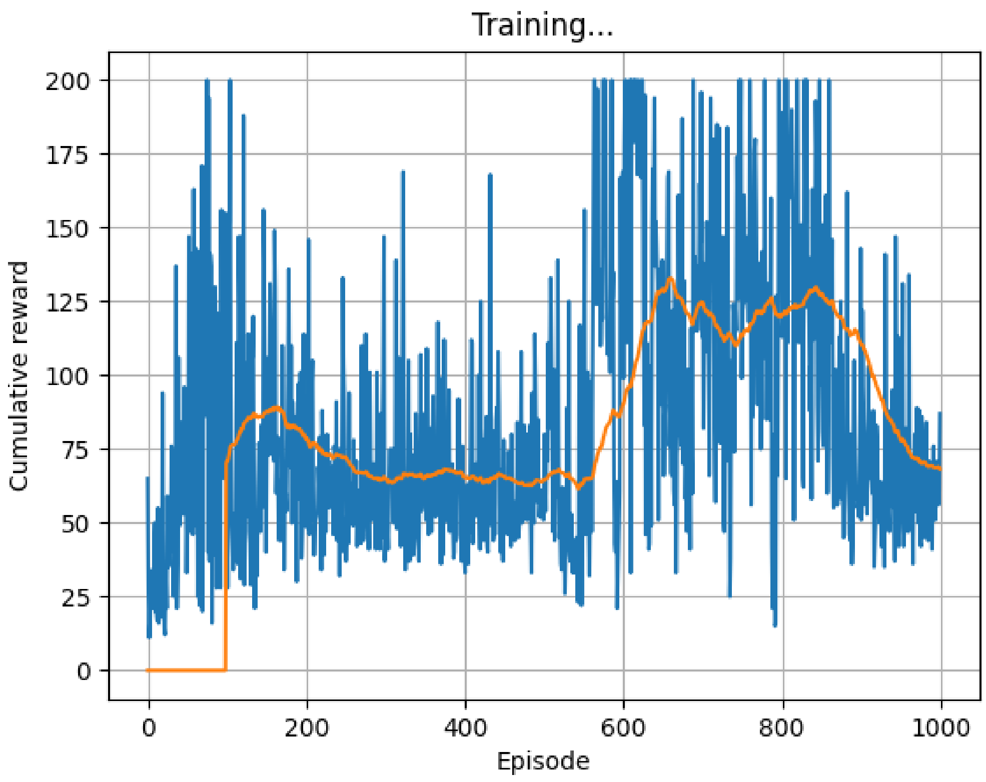


Figure 3: Training performance using handcrafted feature vector $\phi(s) = [s, |s|]^T$.

Source files: main_task2.py, rbf_agent_task2a.py, rbf_agent_task2b.py

Question 2

Figure 2 (from ex4.pdf) 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

How does the experience replay affect the learning performance?

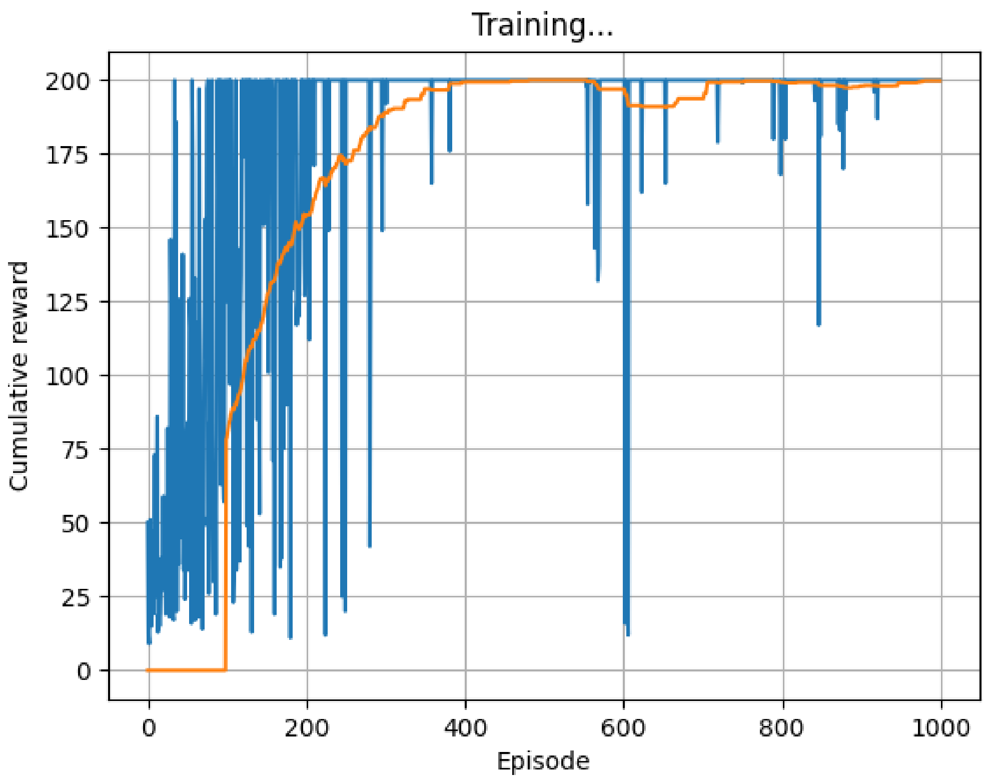


Figure 4: Training performance using radial basis function representations.

The experience replay clearly accelerates convergence to the optimum training performance in all cases presented in Figure 2. This is also quite understandable because the experience replay uses up to 32 times more data for training in the considered CartPole case.

Question 2.2

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 $\phi(s) = [s_x, s_{\dot{x}}, \cos(s_\theta), \sin(s_\theta), s_{\dot{\theta}}]^T$.

The RBF kernel representation of non-linear features comprises 230 features that is clearly superior to the hand-crafted features of type $\phi(s) = [s, abs(s)]$ with 8 features. This simple feature model with 8 features may not take into account of the feature dependencies in the best possible way, which results in poorer performance.

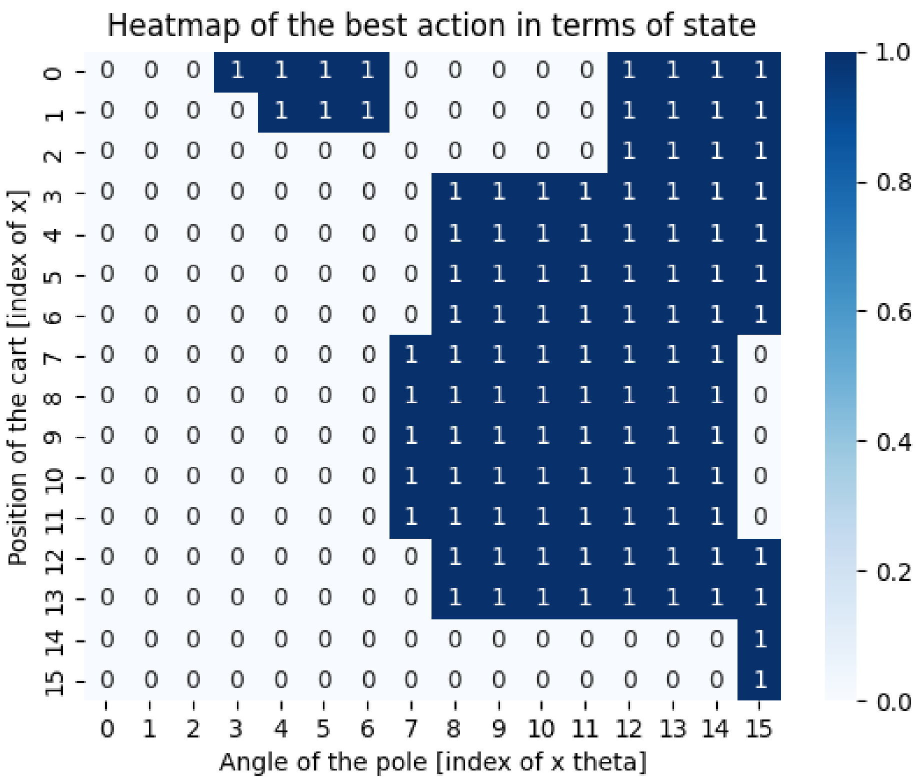
Question 2.3

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

The grid based method are not very sample efficient compared to any of the function approximation methods because, e.g. in the CartPole environment, they require thousands of episodes for convergence, which is achieved in hundreds of episodes with function approximation.

Task 3

The best action in terms of x and θ for $\dot{x} = 0$ and $\dot{\theta} = 0$ learned with RBF with experience replay is presented below for Task 3 in Figure 5. The state-space was discretized for x and θ using the default values of Ex. 3: $[x_{min}, x_{max}] = [-2.4, 2.4]$ and $[\theta_{min}, \theta_{max}] = [-0.3, 0.3]$.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	1	1	1	1	0	0	0	0	0	1	1	1	1
1	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	1
2	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
5	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
6	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
7	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0
8	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0
9	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0
10	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0
11	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0
12	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
13	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 5: Heatmap for the best action in terms of x and θ.

Source files: main_task3.py

Task 4

Training performance using DQN implementation in CartPole environment is presented below for Task 4 in Figure 6.

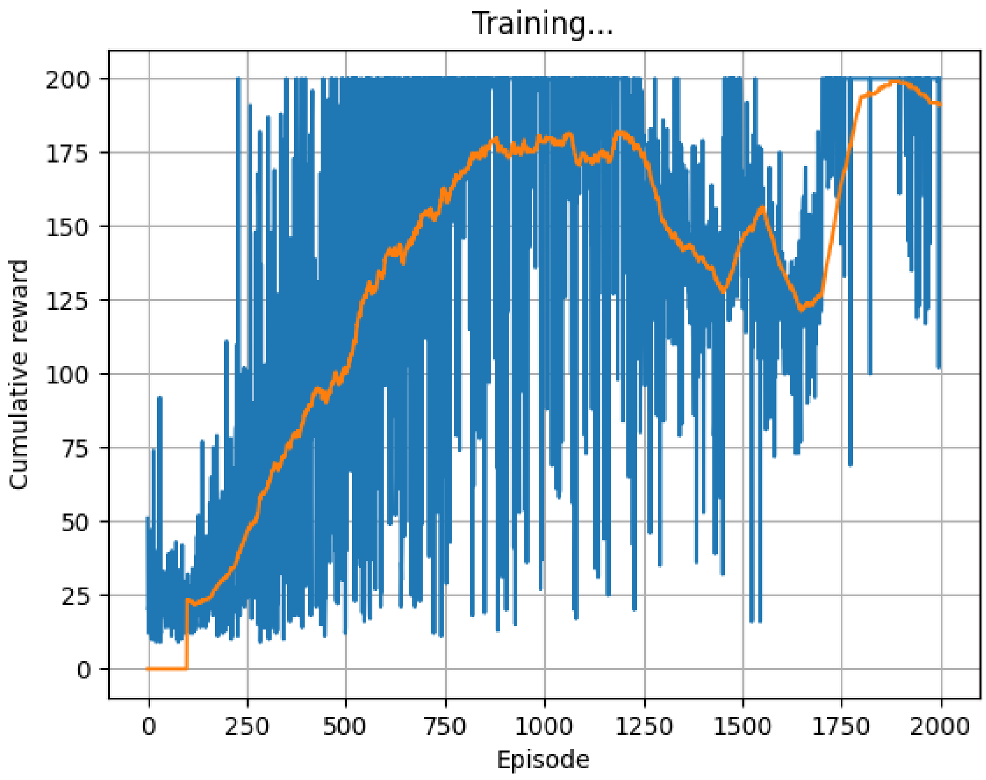


Figure 6: Training performance using DQN implementation in CartPole environment.

The results of the LunarLander environment were unfortunately not ready for the report.

Source files: main_task4.py, dqn_agent_task4.py

Question 3.1

Can Q-learning be used directly in environments with continuous action spaces?

Q-learning can be applied to both discrete and continuous spaces. The obvious approach is to discretize state space, and for particularly large state spaces, some form of functional approximation (such as via a neural network) can be used for the action value.

Question 3.2

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?

Evaluating the maximum in the Q learning equation can become problematic and less accurate for large continuous action spaces. One solution might be to use some actor-critic algorithms.