Deep Reinforcement Learning

Assignment 3

Date: 17.11.2024

Juri Hößelbarth	hoesselbarth@uni-muenster.de	420 924
Mariia Schaak	m.schaak@uni-muenster.de	512 902
Janes-Luca Materne	jmaterne@uni-muenster.de	507 180

Task 1.

a) We implemented the Generalized Policy Iteration according to the book by Sutton&Barto. Unfortunately, our implementation seems flawed, as it does not calculate sensible values. Nevertheless, we attached our code in `task31.ipynb`.

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$; $V(terminal) \doteq 0$

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \left[r + \gamma V(s')\right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma V(s')\right]$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Abbildung 1: Generalized Policy Iteration (source: Deep Reinforcement Learning by Sutton & Barto)

Task 2.

a) The random policy is not always policy suitable due to slow learning as it does not leverage any prior knowledge about the environment, leading to slow performance and failing to take advantage of known good actions, thus lacking exploitation.
The initial policy should have the following characteristics:

- encourage exploration to gather experience, but keep balance between exploration and exploitation
- need to be stable and not lead to immediate termination of the episodes
- should allow learning and improvement over time, avoiding complete random behaviour

The initial policy for `Frozen Lake` environment could be an ϵ-greedy policy to balance exploration and exploitation or heuristic-based policy like moving always move down or right toward the goal.

Task 3.

a)
Conceptualization TD Learning combines characteristics of both Monte Carlo (the sampling) and Dynamic Programming (the bootstrapping). Like Monte Carlo Learning, TD Learning uses a sequence of state transitions to

predict a states return. Updates are available with every step, whilst Monte Carlo Learning only provides results at the end of an episode (which can be very long). This leads to faster convergence and a lower memory footprint. Like Dynamic Programming, TD Learning bootstraps itself by regularly updating value functions depending on the states neighbours, using a Bellman equation. The Sutton Barto describe this as "learning to guess from a guess".

Visualization The heatmap 2 graphically represents the estimated values for each state in the `Frozen Lake` environment. The values of all terminal states (holes and goal state are zero). Blue colors shows the lower state values indicating the less promising states, which are located further from reward. Red color represent the high state values, indicating more promising states, that are located closer to the goal and are more likely to gain the reward.

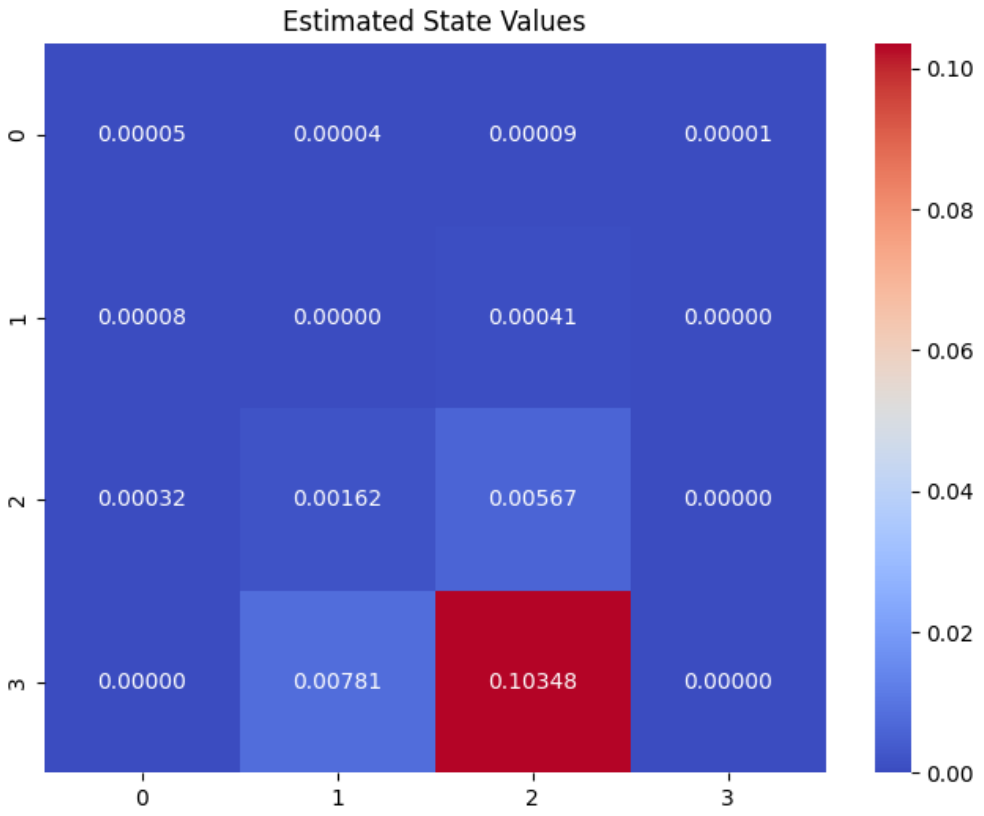

Abbildung 2: Estimated values

Analysis Key advantages of TD learning compared to Monte Carlo methods in regards to sample efficiency are faster learning through incremental updates and faster convergence due to ability to learn from partial experience. In terms of adaptability, TD are less susceptible to problems caused by specific action sequence.

b) We implemented the SARSA algorithm using the ϵ-greedy policy to balance exploration and exploitation. The figure 3 illustrates different scenarios with different exploration parameters that affects learning speed and achieved reward. High ϵ allows the agent to explore more, leading to faster discovery of good policy and getting higher rewards. Low ϵ prevent agent from discovering better option, that why we do not see improvement over time.

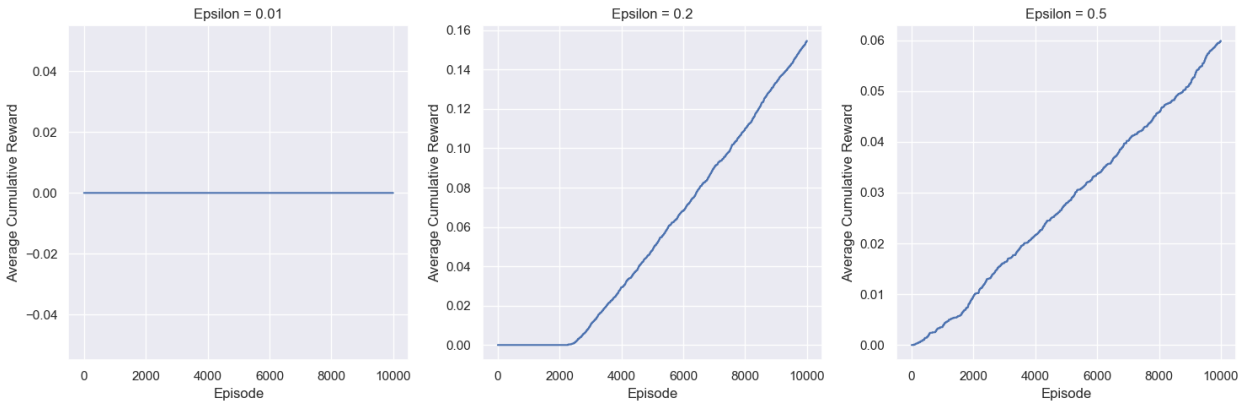

Abbildung 3: Average cumulative reward over the episodes