Prof. Dr. Leif Döring — Reinforcement Learning

Sara Klein, Benedikt Wille

6. Exercise Sheet

1. Policy evaluation

Consider Algorithm 8 from the lecture. In Theorem 3.3.2 we proved convergence for this algorithm if $\gamma < 1$. Now assume $\gamma = 1$ and set $\Delta = 2\epsilon$ in the initialisation and choose termination condition $\Delta < \epsilon$. Give an example such that Algorithm 8 does not converge using $\gamma = 1$. You are allowed to initialise the value function V arbitrarily.

2. Convergence of the in-place policy evaluation algorithm

Recall Algorithm 9 from the lecture. We aim to prove convergence of the algorithm (without termination) to V^π. Therefore, label the state space $\mathcal{S}$ by $s_1, \dots, s_K$ and define

$$T_s^\pi V(s') = \begin{cases} T^\pi V(s) & : s = s' \\ V(s) & : s \neq s' \end{cases}$$

Define the composition $\overline{T}^\pi : U \to U$, $\overline{T}^\pi(v) := \left(T_{s_K}^\pi \circ \cdots \circ T_{s_1}^\pi\right)(v)$ on the space of all functions $U = \{u : \mathcal{S} \to \mathbb{R}\}$ equipped with the supremum-norm.

a) Argue why $\overline{T}^\pi$ is different from the Bellman operator T^π.

b) Show that V^π is a fixpoint of the operator $\overline{T}^\pi$.

c) Prove that $\overline{T}^\pi$ is a contraction on $(U, ||\cdot||_\infty)$.

3. Exact policy iteration

Consider two types of costumers, L for low and H for high, shopping in a shopping center. Each quarter the manager divides all costumers into these classes based in their purchase behavior in the previous quarter. The manager wishes to determine to which classes of costumers he should send quarterly catalogs. Sending a catalog costs him \$15 per costumer. If a costumer received a catalog at the beginning of the quarter and is in class L at the subsequent quarter, then the expected purchase is \$20, and \$10 if he did not receive a catalog. If a costumer is in class H at the subsequent quarter and received a catalog, then the expected purchase is \$50, and \$25 if he did not. The decision weather or not to send a catalog to a customer also affects the customer's classification in the subsequent quarter: If a costumer is in class L at the start of the present quarter, then the probability to stay in class L in the subsequent quarter is 0.3 if he receives a catalog and 0.5 if he does not. For the class H costumer the probability to stay in H for the subsequent quarter is 0.8 if he receives a catalog and 0.4 if he does not.
Assume that the discount rate is 0.9 and the manager wants to maximize the expected total discounted reward.

a) Formulate this problem as discounted infinite-horizon Markov decision model.

b) What is the expected one-step reward for every state-action pair? Define the stationary policy which has greatest one-step reward.

c) Find an optimal policy using the greedy exact policy iteration (algorithm 10) to find the optimal policy. Start with the stationary policy from b).

4. Programming Task: Bandits - Will be discussed in class!

The aim of this exercise is to play around a little with the different algorithms for bandits that we saw in the lecture in different settings. This is a very long and comprehensive exercise, but you don't need to write code by yourself if you don't want to, instead you are highly encouraged to use ChatGPT or other tools for generating your code. It is every easy - try it! Since the focus does not lie on implementing but rather on the gained insights we will discuss the results of this exercise in the tutorial session.

a) Implement a Gaussian bandit where the arms have random means that are standard normally distributed, and a Bernoulli bandit where the arms have random means p that are uniformly distributed between 0 and 1. Additionally, implement a Gaussian bandit where you draw standard normally distributed means but then fix the mean of the highest arm and replace the other means in descending order by values decreasing by a fixed reward gap Δ, i.e. if the arm with the highest mean has value μ^*, the mean of the k-th biggest arm gets replaced by $\mu^* - k\Delta$.

b) Implement the following bandit algorithms: Explore-then-committ, ϵ-greedy, UCB, and Boltzmann exploration.

c) Implement the policy gradient method with softmax parametrization starting with a uniform policy, i.e. a vector $\theta = k \cdot (1, \dots, 1)$.

d) Play each bandit with each algorithm $N = 5000$ times using a time horizon of $n = 500$. Vary the reward gaps by choosing $\Delta = 0.1, 0.3,$ and 0.5 and the number of arms by choosing $K = 5, 10,$ and 20. Try to find the „best“parameters for each of the algorithms. Then compare the outcomes in the following way:

   i) Plot the regrets,

   ii) Plot the average probabilities of choosing the different arms over the time horizon,

   iii) For the last timestep, compute the average of the probabilities for playing each arm in the next round,

   iv) For the last timestep, compute what the algorithm makes out to be the mean of the arms against their real means and add the confidence bounds, and

   v) Plot how many times on average each algorithm chooses the best arm in total and plot the average (cumulative) hitting times of the best arms over the time horizon.

   What do you observe? Think about possible implications in the reinforcement learning setting and especially how the different goals of bandits and reinforcement learning impact what a good learning strategy is.

5. **Programming Task*: Policy Iteration**

In this programming task we want to program a first game, as well as an agent that plays the game. In addition, we want to use the algorithms from the lecture to optimize the policy of the agent.

(a) Implement the Grid World game from the lecture (example 1.1.9). Create a class that can be used to play the game. This code might be helpful.

(b) Implement a class player that has a decision rule, such that the game from the previous task can be played.

(c) Use the generalized policy iteration paradigm (algorithm 12) to find the optimal policy for the game. For policy evaluation use Algorithm 8 and for policy improvement use Algorithm 10.

6. **Programming Task*: Monte Carlo generalised ε-greedy policy iteration**

In this exercise, we will use a Monte Carlo estimator to perform policy iteration. After Grid World has been a first example for a Markov decision process, we will now deal with more complex examples.

a) Implement the ice vendor (Example 3.1.8) as a Markov decision process.

b) Use the policy iteration from the last lecture to get the optimal decision rule for the iceman.

c) Implement the Monte Carlo generalised ε-greedy policy iteration (algorithm 18) to get the optimal decision rule for the iceman. Additionally, use different $\varepsilon > 0$ parameters as well as a sequence $\varepsilon_n \downarrow 0, n \to \infty$.