FAKULTÄT FÜR WIRTSCHAFTSINFORMATIK
UND WIRTSCHAFTSMATHEMATIK

UNIVERSITÄT MANNHEIM

Prof. Dr. Leif Döring
Daniel Schmidt, Benedikt Wille

Reinforcement Learning

5. Exercise Sheet

16.03.2026

For this assignment, knowledge from lectures 1 to 12 is assumed.

1. Policy evaluation

Consider Algorithm 7 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 7 does not converge using $\gamma = 1$. You are allowed to initialise the value function V arbitrarily.

2. Convergence of the iterative policy evaluation with totally asynchronous updates

Recall Algorithm 8 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: Algorithms using game dynamics

The aim of this exercise is to implement algorithms that leverage explicit knowledge of game dynamics to evaluate policies and compute value functions using Bellman Operators. Thus, implement the following algorithms:

- Value iteration (e.g. Algorithm 6).
- Iterative policy evaluation (e.g. Algorithm 7).
- Policy iteration/Actor-critic (e.g. Algorithm 10).

For each algorithm, implement two versions: one estimating the value function V and another estimating the state-action value function Q. Use the corresponding Bellman Operators to adapt each algorithm accordingly. Additionally, provide an option for fully asynchronous execution (cf. Algorithm 8), where updates happen immediately without saving interim values. In the actor-critic setting, include configurations for both fixed and functionally dependent steps for the actor's updates. Implement all algorithms such that they can be performed either for a determined amount of steps or until a certain error tolerance is reached.
Hint: Switching to asynchronous updates requires directly updating V or Q without temporary storage. To adapt algorithms for Q instead of V, simply replace the Bellman operator for V with the one for Q within the update step.

5. *Programming task: Backpropagation

For both environments from the previous sheet, design and implement a visual representation of policies and value functions to track how they evolve throughout the algorithmic steps compared to the optimal solution. Apply this to a 4×4 Grid World with a Goal state ($R = 10$) in the bottom right and a Start state in the top left (default reward 0). Use these visualizations to analyze stochastic control and policy evaluation, and reflect on what backpropagation might refer to in this context.

6. *Programming task: Dynamic Programming

Implement policy evaluation and optimal control algorithms for finite-time MDPs (Algorithms 12 and 13). For both the Grid World and Multi-step Bandit from the previous sheet, consider

how the structure of these environments could be leveraged to accelerate the algorithms and try implementing versions that use this knowledge.
Hint: For speeding up the algorithm, your insights from the task before may be helpful.

7. *Programming task: Convergence rates

Empirically verify the convergence rates provided in the lecture for selected algorithms. Compare the empirical error rates of all implemented algorithms through graphical analysis.

8. *Programming task: Finite and infinite time MDPs

Analyze the theoretical differences between finite-time and infinite-time MDP settings using the GridWorld from Exercise 5 with varying default rewards of $-1, 0$, and 1. Compare these insights to empirical observations.

The solution to the theoretical exercises will be discussed in the exercise class in B2 on March 26, 2026, at Mathelounge in B6 B301.