FAKULTÄT FÜR WIRTSCHAFTSINFORMATIK
UND WIRTSCHAFTSMATHEMATIK

UNIVERSITÄT MANNHEIM

Prof. Dr. Leif Döring
Daniel Schmidt, Benedikt Wille

Reinforcement Learning

4. Exercise Sheet

09.03.2026

For this assignment, knowledge from lectures 1 to 10 is assumed.

1. Bellman expecation operator.

Recall the Bellman expectation operator for a stationary policy $\pi \in \Pi_s$:

$$\begin{aligned}(T^\pi u)(s) &= \sum_{a\in\mathcal{A}} r(s,a)\pi(a\,;\,s) + \gamma \sum_{s'\in\mathcal{S}} \mathbb{P}^\pi(S_1 = s' | S_0 = s, A_0 = a)u(s') \\ &= \sum_{a\in\mathcal{A}} \pi(a\,;\,s)\Big(r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s';\,s,a)u(s')\Big)\end{aligned}$$

Show that we can rewrite the fixed point equation in vector notation, i.e. check that indeed $T^\pi V = V$ is equivalent to $r_\pi + \gamma P_\pi V = V$, where

$$\begin{aligned}P_\pi &= \Big(\sum_{a\in\mathcal{A}} \pi(a\,;\,s)p(s'\,;\,s,a)\Big)_{(s,s')\in\mathcal{S}\times\mathcal{S}} \\ r_\pi &= \Big(\sum_{a\in\mathcal{A}} \pi(a\,;\,s)r(s,a)\Big)_{s\in\mathcal{S}}.\end{aligned}$$

2. Markov Decision Process 1

A rental car agency serves two cities with one office in each. The agency has M cars in total. At the start of each day, the manager must decide how many cars to move from one office to the other to balance stock. Let $f_i(q)$ for $i = 1, 2$ denote the probability that the daily demand for cars to be picked up at city i and returned to city i equals q. Let $g_i(q)$, $i = 1, 2$, denote the probability that the daily demand for "one-way" rentals from city i to the other equals q. Assume that all demands are independent. Assume that all rentals are for one day only, that it takes one day to move cars from city to city to balance stock, and, if demand exceeds availability, customers go elsewhere. The economic parameters include a cost of K per car for moving a car from city to city to balance stock, a revenue of R per car for rentals returned to the rental location and R per car for one-way rentals.
Formulate this problem as discounted infinite-horizon Markov decision model.

3. Markov Decision Process 2

Two identical machines are used in a manufacturing process. From time to time, these machines require maintenance which takes three weeks. Maintenance on one machine costs c, per week while maintenance on both machines costs c_2 per week. Assume $c_2 > 2c$. The probability that a machine breaks down if it has been i periods since its last maintenance is p_i,with p_i non-decreasing in i. Maintenance begins at the start of the week immediately following breakdown,

but preventive maintenance may be started at the beginning of any week. The decision maker must choose when to carry out preventive maintenance, if ever, and, if so, how many machines to repair. Assume the two machines fail independently.

a) What is the significance of the condition $c_2 > 2c$?

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

c) Provide an educated guess about the form of the optimal policy when the decision maker's objective is to minimize expected operating cost.

4. Banach fixed-point Theorem

Prove Banach fixed-point theorem (Theorem 3.1.17 in the lecture notes).

5. *Programming task: Two environments

The aim of this task is to implement two environments: Grid World (as introduced in the lecture) and a Multi-Step Bandit. Each environment should be encapsulated in a class with:

- A function for game dynamics. Additionally to the next state and reward obtained it should also return if the game terminated or not.
- A function estimating state-action rewards using a Monte Carlo estimator.
- A reset mechanism: after reaching a terminal state, restart from the initial state.

For future exercises, ensure your implementation includes transition probabilities, start state, terminal states, allowed actions, and expected rewards as class attributes. Additionally, for Grid World, implement a function that visualizes a given policy. Your implementation should support at least normally and binomially distributed random rewards.

a) Implement the standard Grid World and variants to explore different effects. Your implementation should allow configuring:
   - Grid size ($m \times n$),
   - Reward structure, assigning deterministic or stochastic rewards to specific cells (e.g., Goal, Default, Bomb).
   - Wall behavior: choose whether actions leading outside the grid result in staying put or are prohibited.
   - Windy environment: with some probability, the agent moves in a predefined unintended direction.
   - Slippery environment: with some probability, the agent moves in an unintended direction adjacent to the chosen one.
   - Random noise: with some probability, the agent moves in a completely random direction, where the probabilities for each directions can be specified.

b) Implement a multi-step bandit with the following structure:

- A starting state s_0 with k available actions (branches).
- Each branch b_i consists of m_i sequential steps ($i = 1, \ldots, k$). The agent transitions deterministically along the branch.
- At each step $s_{b_i,j}$, the agent chooses from $k_{b_i,j}$ possible actions ($j = 1, \ldots, m_i$).

Allow specifying default rewards and custom deterministic or random rewards per action at each state.

Hint: RL environments are very similar to bandits but require an additional state input and must return the next state along with the reward for each action.

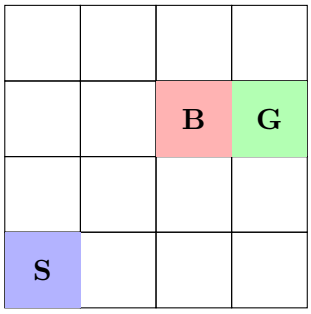

Figure 1: 4×4 Grid World environment. The agent starts at S (bottom-left). The goal G (top-right) gives a reward upon reaching it. Stepping on B (bomb) results in a penalty. The agent can move in four directions unless restricted by walls.

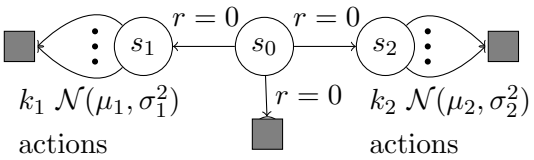

Figure 2: Multi-step bandit environment where the agent starts at the initial state s_0 and can choose between three branches: left, right, or down. The downward branch leads to a terminal state with zero reward. The left and right branches each consist of two steps. The first step yields zero reward, while the second step offers k_i action choices, each associated with a reward sampled from a normal distribution $\mathcal{N}(\mu_i, \sigma_i^2)$. The indices $i = 1, 2$ correspond to the left and right branches, respectively.

6. *Programming task: Hard-coding Policy evaluation and the optimal policy

Develop a hard-coded method to determine the optimal policy for two settings: a vanilla Grid World (without wind, slipping, or noise, only start and goal) and an arbitrary multi-step bandit. Implement this approach twice—first using expected rewards, then using Monte Carlo (MC) estimation. Extend your implementation to evaluate arbitrary policies by calculating their value functions, again with and without MC estimators. Consider the algorithmic complexity, identify „worst-case“-environments, and think about how long it would take you to find the best policy knowing the reward structure while comparing your approach to your algorithm's.

7. *Programming task: The best, the worst, uniform, and random policies

Analyze the impact of wind, slipping, and random noise on a 4×4 Grid World with arbitrary layouts, ensuring at least one bomb state. Identify what you consider the best and worst policies for each case and test them against randomly generated policies and a uniformly random policy. Evaluate the discounted rewards after $t = 16$ rounds and, for stochastic policies, average over $N = 1000$ runs to compare performance.

Hint: A similar functional approach as in the bandit setup can be applied, only that now you need to keep track of the current state. The policy selects an action based on the current state, after which the environment returns a reward and the next state. This next state is then used to determine the subsequent action. Ensure that the policy-environment interaction follows this sequence in each step to accurately reflect the dynamics of the multi-state environment.

The solution to the theoretical exercises will be discussed in the exercise class in B2 on March 23, 2026, at Mathelounge in B6 B301.