FAKULTÄT FÜR WIRTSCHAFTSINFORMATIK
UND WIRTSCHAFTSMATHEMATIK

UNIVERSITÄT MANNHEIM

Prof. Dr. Leif Döring
Benedikt Wille

Reinforcement Learning

6. Exercise Sheet

25.03.2025

For this assignment, knowledge from lectures 1 to 14 is assumed.

1. Proof of Lemma 3.4.6 for T-step MDPs

Prove Lemma 3.4.6 from the lecture by comparing with the discounted counterpart.
The following holds for the optimal time-state value function and the optimal time-state-action value function for any $s \in \mathcal{S}$:

(i) $V_t^*(s) = \max_{a \in \mathcal{A}_s} Q_t^*(s,a)$ for all $t < T$,

(ii) $Q_t^*(s,a) = r(s,a) + \sum_{s' \in \mathcal{S}} p(s'\,;\, s,a) V_{t+1}^*(s')$ for all $t < T$

In particular, V^* and Q^* satisfy the following Bellman optimality equations (backwards recursions):

$$V_t^*(s) = \max_{a \in \mathcal{A}_s} \Big\{ r(s,a) + \sum_{s' \in \mathcal{S}} p(s'; s, a) V_{t+1}^*(s') \Big\}, \quad s \in \mathcal{S},$$

and

$$Q_t^*(s,a) = r(s,a) + \sum_{s' \in \mathcal{S}} p(s'; s, a) \max_{a' \in \mathcal{A}_{s'}} Q_{t+1}^*(s', a'), \quad s \in \mathcal{S}, a \in \mathcal{A}_s,$$

for all $t < T$.

2. Example: T-step MDPs

Recall the Ice Vendor example from the lecture. Assume the maximal amount of ice cream is $m = 3$ and the damand distribution is given by $\mathbb{P}(D_t = d) = p_d$ with $p_0 = p_2 = \frac{1}{4}, p_1 = \frac{1}{2}$. Suppose the revenue function f, ordering cost function o and storage cost function h are given by

$$f : \mathbb{N}_0 \to \mathbb{R},\ x \mapsto 9x,$$
$$o : \mathbb{N}_0 \to \mathbb{R},\ x \mapsto 2x,$$
$$h : \mathbb{N}_0 \to \mathbb{R},\ x \mapsto 2 + x.$$

a) Set up the transition matrix $p(s_{t+1}; s_t, a_t)$ in a table, such that every $s_t + a_t$ maps to the probability to land in s_{t+1}, and the reward function $r(s_t, a_t, s_{t+1})$ for this example.

b) Calculate the expected reward $r(s,a)$ for every state action pair. Can you guess an optimal strategy for a one time step MDP?

c) Suppose now you can play a 3-step MDP, hence you can order ice cream 4 times in $t = 0, 1, 2$. What is the optimal strategy for this finite time horizion MDP? Calculate the optimal state value, state-action value functions and the optimal policies using the optimal control algorithm from the lecture.
Hint: Use backward induction.

3. First visit Monte Carlo

Recall the first visit Monte Carlo Algorithm (14) from the lecture notes. Rewrite the estimate $V_n(s_t)$ to argue how we can apply the law of large numbers to show convergence (Hint: Use the strong Markov property).
Now consider the same algorithm without the if-condition in the for-loop. This algorithm is called every visit Monte Carlo algorithm (see Algorithm 1). Argue why we cannot apply the law of large numbers.

Data: Policy $\pi \in \Pi_S$, initial condition μ
Result: Approximation $\tilde{V} \approx V^\pi$
Initialize $V_0 \equiv 0$ and $N \equiv 1$
$n = 0$
while *not converged* **do**
  $n = n + 1$
  Sample $T \sim \text{Geo}(1 - \gamma)$.
  Sample s_0 from μ.
  Generate trajectory $(s_0, a_0, r_0, s_1, ...)$ until time horizon $2T$ using policy π.
  for $t = 0, 1, 2, ..., T$ **do**
    $v = \sum_{k=t}^{T+t} r_k$
    $V_n(s_t) = \frac{1}{N(s_t)+1} v + \frac{N(s_t)-1}{N(s_t)} V_{n-1}(s_t)$
    $N(s_t) = N(s_t) + 1$
  end
end
Set $\tilde{V} = V_n$.

Algorithm 1: Every visit Monte Carlo policy evaluation of V^π

4. *Programming task: Sample-based algorithms

The aim of this task is to implement the main algorithms from chapter 4 of the lecture. To this end, include the following algorithms:

a) The first visit Monte Carlo policy evaluation methods (Algorithms 14 and 15) as well as every visit versions for both V^π and Q^π (cf. Algorithm 1).

b) The totally asynchronous policy evaluation methods (Algorithms 17 and 18).

c) The Q-learning algorithm (Algorithm 18) with fixed behaviour policies.

To support these implementations, write a function that schedules the chosen stepsizes, incorporating both constant stepsizes and stepsizes decreasing with a predetermined rate function.

5. *Programming task: The cost of not knowing the game

The aim of this task is to compare implemented algorithms with their counterparts that assume knowledge of the game dynamics and among each other. For Q-learning, use several different exploration methods. Implement a mechanism to evaluate the current policy after m steps of the algorithm: pause training, evaluate the policy for k timesteps, and then resume training. Compare the algorithms in terms of average scores, correct action rates (the percentage of actions chosen correctly), Q-function values at the start state, and runtime. Additionally, investigate whether the backpropagation effect occurs in sample-based algorithms.

6. *Programming task: Do we need 1/n?

Experiment with different stepsize schedules for Q-learning across various Grid World environments. Investigate the short- and long-term effects of these schedules and analyze why certain behaviors emerge. Develop an intuition for the appropriate stepsize schedule needed for different environmental variations, considering factors such as convergence speed, stability, and final performance. Lay a special focus on comparing your results with the rate of $\frac{1}{n}$ that is always used in convergence proofs.

The solution to the theoretical exercises will be discussed in the exercise class in B5 on April 01, 2025, at Mathelounge in B6 B301.