Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Matthias Schubert
Maximilian Bernhard, Niklas Strauß

Munich, 20.01.2025

Deep Learning and Artificial Intelligence

WS 2024/25

Exercise 11: Modelfree Reinforcement Learning

Exercise 11-1 RL Theory Questions

(a) In Reinforcement Learning, what do the terms 'prediction' and 'control' refer to?

Detailed Solution

Prediction = policy evaluation: Computing the state-value function v_π for a policy π.
Control = approximate optimal policies

(b) What are the differences between model-free and model-based methods? What do they have in common? Which category do Dynamic Programming (DP), Temporal-Difference-Learning (TD) and Monte Carlo (MC) -methods belong to?

Detailed Solution

- Model-based: Methods for solving RL problems that use a model of the environment (anything an agent can use to predict how the environment will respond to its actions, e.g. transition probabilities or next states and rewards).
  Model-free: No model is needed (for the updates of the value estimates).
- Both compute value functions and are based on looking ahead to future events, computing backed-up values and using them as targets for an approximate value function.
- DP-methods are model-based since one has to know the transition probabilities $p(s' \mid s, a)$ in order to calculate the state-value estimate as an expected value over all possible next states (see exercise 11-2 a))
  TD-methods and MC control methods are model-free. Both methods just update their value estimates based on real experience (samples).

(c) Write down the Bellman equations for optimal state values (utilities), denoted $u_*(s)$, and optimal state-action values, denoted $q_*(s, a)$. How are $u_*(s)$ and $q_*(s, a)$ related to each other?

Detailed Solution

The state and state-action values for a given policy π are defined as:
$u_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$ and
$q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$ respectively, where G_t is the discounted return.

The Bellman equations for $u_*(s) = \max_\pi u_\pi(s)$ and $q_*(s, a) = \max_\pi q_\pi(s, a)$ are:

$$u_*(s) = \max_a \sum_{s'} p(s' \mid s, a)[R(s') + \gamma\, u_*(s')]$$

$$q_*(s, a) = \sum_{s'} p(s' \mid s, a)[R(s') + \gamma \max_{a'} q_*(s', a')]$$

They are related to each other by: $u_*(s) = \max_a q_{\pi_*}(s, a)$, i.e., the optimal value for state s corresponds the optimal state-action value when taking the best action in state s.

(d) What does on- and off-policy learning mean? Name an example of each.

Detailed Solution

On-policy methods evaluate or optimize the policy that is used to make the decisions (sampling), e.g. Sarsa (on policy TD-control)
Off-policy methods evaluate or optimize a *target policy* by learning from samples generated by a different *behavior policy*, e.g off-policy Monte Carlo/TD- prediction/control via importance sampling, Q-learning (off-policy TD-control).

(e) Why is Q-learning an off-policy method? What is the difference to Sarsa?

Detailed Solution

The update rule for Q-learning is

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)],$$

i.e. Q-learning always uses the best Q-value for the update independent of the action that the policy would have actually chosen.

Whereas Q-learning backs up the best Q-value from the state reached in the observed transition, Sarsa waits until an action is actually taken and backs up the Q-value for that action. For a greedy agent that always takes the action with the best Q-value, the two algorithms are identical. When exploration is happening, however, they differ significantly.

(f) When do we need exploration and why? Why is there a trade-off between exploitation and exploration?

Detailed Solution

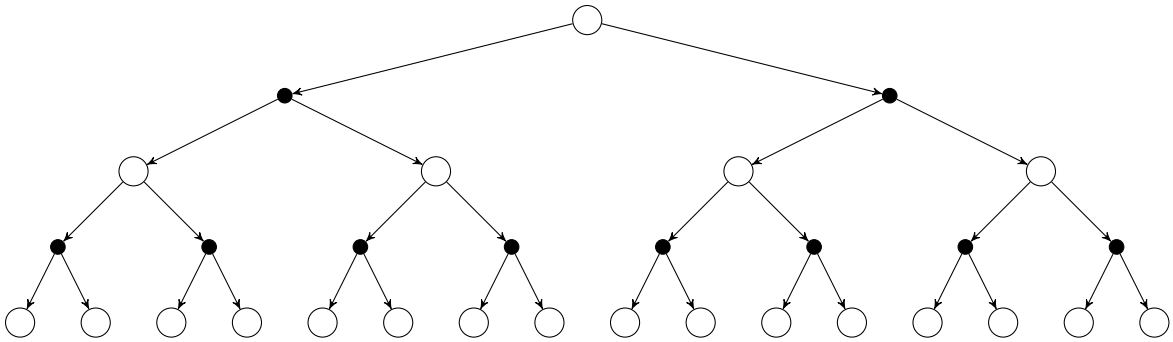
Exploration is necessary when the environment is unknown and the agent has no idea about the states and the effects of its actions.

To obtain a lot of reward, an RL agent must prefer actions that it has tried in the past and found to "good" in regards of reward. However, to discover such actions, it has to try actions that it has not selected before. The agent has to exploit what it has already experienced in order to obtain reward, but it also has to explore in order to make better action selections in the future.

Exercise 11-2 Backup Strategies

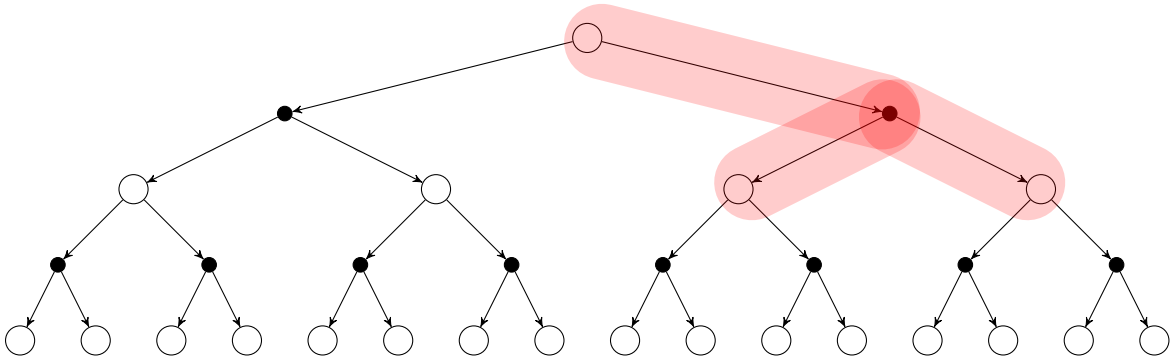
In the lecture you learned three different backup strategies for the Bellman equation (Dynamic Programming (DP), Monte Carlo (MC) and Temporal Difference Learning (TD)).

(a) Assume we have an MDP with exactly two steps (two times an action is performed). The figure below visualizes this MDP. An unfilled circle represents a state and a filled circle represents an action respectively.
For each of the different backup strategies, mark the pathes that are used. For DP, assume a fixed policy $\pi(S_t)$ in the first step (instead of max). For MC and TD, just choose arbitrary actions. Also write down the formulas for these updates.



Detailed Solution

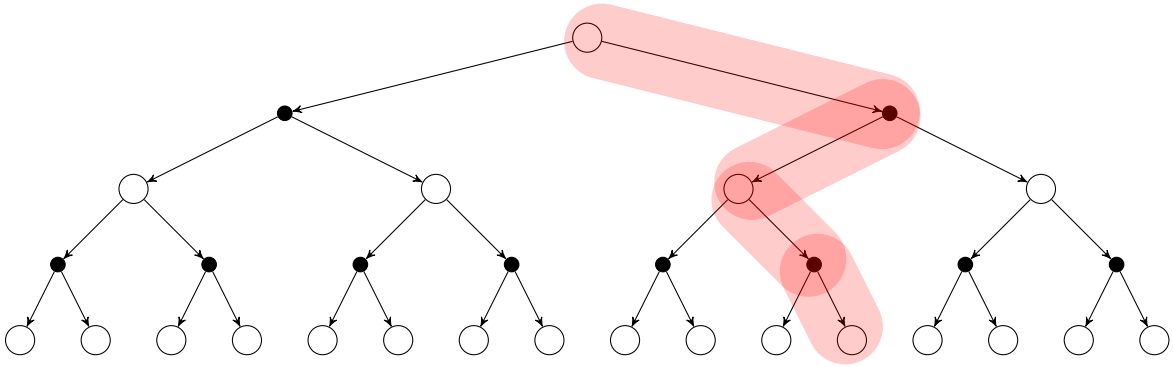
Dynamic Programming:



$$\begin{aligned} U(S_t) \leftarrow & \mathbb{E}[R_{t+1} + \gamma U(S_{t+1}))] \\ & = \sum_{s'} p(s' \mid S_t, \pi(S_t))[R_{t+1} + \gamma U(s')] \end{aligned}$$

Important: Note that R_{t+1} can depend just on the outgoing state S_t (in which case it can be taken out of the sum over s' as a constant) or on the state we arrive at $S_{t+1} = s'$ or, in some cases, even on the combination of state and action S_t, A_t. This is a matter of how the underlying MDP is defined and has to be given.

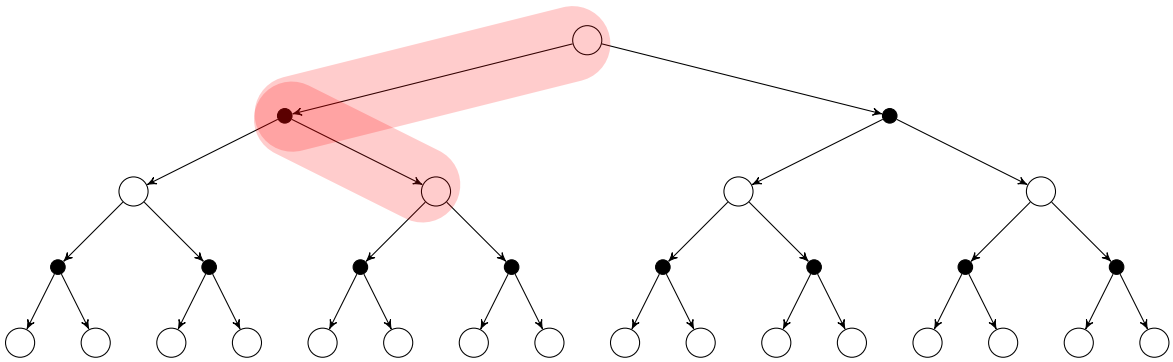
Monte Carlo:



$$U(S_t) \leftarrow U(S_t) + \alpha(G_t - U(S_t))$$

$$\text{where} \quad G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Temporal Difference Learning:



$$U(S_t) \leftarrow U(S_t) + \alpha(R_{t+1} + \gamma U(S_{t+1}) - U(S_t))$$

(b) How do the three backup strategies compare regarding variance, efficiency, necessity of a model and bias?

Detailed Solution

Dynamic Programming:

+ low variance
+ efficient
- model needed
- high bias

Monte Carlo:

- high variance
- low efficiency
+ no model needed
+ bias-free

Temporal Difference:

+ low variance
+ efficient
+ no model needed
- high bias

Exercise 11-3 Model-free RL in Python

On moodle course page you can find a Jupyter notebook file with a programming exercise for model-free reinforcement learning. Please follow the instructions in the notebook.