Question 1. (8 points)

Consider a Markov decision process.

1. (3 points) Explain why a fixed (independent of $x_1, x_2, \ldots$) sequence of actions as $y_1, y_2, \ldots$ ($y_k \in Y$) does not solve a Markov decision process, i.e. cannot guarantee optimality in reinforcement learning. In which field of AI would a fixed sequence of actions be an appropriate solution? Where is the boundary between game theory and reinforcement learning?

   Answer:

   No fixed sequence of actions is guaranteed to be optimal in MDPs as the environment is stochastic and the current action must be chosen in dependence to the observed state. A fixed sequence of states is a valid solution for classical (i.e., not probabilistic) planning. The MDPs and RL setting can be seen, therefore, as a generalization of the planning. The game theory field is generalization of the MDPs and RL as it includes possibility for multiple agents and not only stochastic environment.

2. (2 points) Recall the value iteration algorithm. This algorithm is based on a general method for solving a set of equations. Give the name for this method and explain how it works in one or two sequences.

   Answer:

   The value iteration algorithm is based on fixed point method (in Czech, *metoda prosté iterace*) for finding a solution of a set of equations. This method is based on iterative application of the Bellman operator on the initial estimate of the utility.

3. (3 points) Recall the policy iteration algorithm. This algorithm is based on another well known concept from machine learning and statistics. Name this algorithm and explain its idea in one or two sequences. Provide an example of other usages of this algorithm in computer science or mathematics.

   Answer:

   The policy iteration algorithm is based on the same idea as the expectation–maximization (EM) algorithm. In this algorithm, we aim on calculating the estimates of the utility $\widehat{U}$, having the max operator in the set of equations. Instead of solving this set of equations directly, we define a set of additional parameters (the policy). Then we iteratively optimize the utility estimates with respect to a fixed policy, followed by a step where we optimize the policy w.r.t. fixed utility estimates. Alternating those two steps is guaranteed to find an optimal policy. The EM algorithm can be found (among others) in Baum-Welsch algorithm for learning hidden Marcov models or estimation of the parameters for a mixture of Gaussians.

Question 2. (5 points)

In the lecture and tutorials, we state the Bellman equations the following way.

$$U(x) = r(x) + \gamma \max_{y \in Y(x)} \sum_{x'} P(x' \mid x, y) U(x').$$

In some literature, you may find under the same name a different equation.

$$U(x) = \max_{y \in Y(x)} \sum_{x'} P(x' \mid x, y)(r(x, y, x') + \gamma U(x'))$$

Explain what is different in those two situations and decide which one is more general than the other.

Answer:

In the first representation, the reward depends only on the current state, in the second the reward depends on the previous state, action taken, and the current state. Assuming that the second two parameters of the reward are irrelevant, i.e.,

$r(x, y, x') = r(x)$, we can rewrite the second representation as

$$\max_{y \in Y(x)} \sum_{x'} P(x' \mid x, y)(r(x) + \gamma U(x')) =$$
$$\max_{y \in Y(x)} \left(\sum_{x'} P(x' \mid x, y) r(x) + \gamma \sum_{x'} P(x' \mid x, y) U(x') \right) =$$
$$\max_{y \in Y(x)} \left(r(x) + \gamma \sum_{x'} P(x' \mid x, y) U(x') \right) =$$
$$r(x) + \gamma \max_{y \in Y(x)} \sum_{x'} P(x' \mid x, y) U(x'),$$

which shows that the second representation is as least as general as the first one. It remains to show that the first representation can model anything that the second one can. The key idea is to modify the state space. We can create a new state for each possible combination of state-action-state by defining $x_{\mathsf{new}} = (x, y, x')$. This inflation of the state space allows us to make reward dependent only on the state. (By the way, the modification is the same if we wanted to translate MDPs of the second-order to MPDs of the first order.) Therefore, both situations are equivalent, and we can deal only with the first one.

Question 3. (12 points)

Despite many reinforcement learning algorithms with additive rewards, it is common to use discounted rewards to model the environment.

1. (1 point) Give the range for the discount factor parameter.

   Answer:

   The discount factor is from $[0, 1]$.

2. (1 point) How does the agent behave when $\gamma = 0$?

   Answer:

   The agent behaves greedily, goes only for the highest reward in the next step.

3. (2 points) Give an example of an environment where a high discount factor is a good choice and why. Do the same for a low discount factor.

   Answer:

   The high discount factor is appropriate when it is irrelevant how long it will take the agent to obtain the reward in a terminal state. For example, when playing chess, it is more relevant whether the agent wins or loses than the number of moves. Therefore, the discount factor will be close to one. If our goal is to get the highest number of wins, then the discount factor will be exactly one.

   A small discount factor might be used in scenarios when the agent is in a very unstable environment. Imagine a robot on Mars that wants to decide whether to send experimental results to Earth. As dust storms are likely, it is wise to send data earlier, meaning low discount factor. Similar results can be achieved by a small negative reward in each non-terminal state.

4. (3 points) In the case of the infinite horizon, discounting the rewards is necessary. Explain why.

   Answer:

   If some of the agent histories are infinite with repeated positive rewards, it may happen that infinite sequences are not comparable as, for example, $1 + 1 + 1 + \cdots$ and $2 + 2 + 2 + \cdots$ have both values ∞. The second one is, however, clearly preferred. Discounting allows us to distinguish between those two as for any $\gamma \in (0, 1)$, the second one has a higher discounted sum.

5. (3 points) Imagine that your fancy reinforcement learning algorithm is not working on a chess game. Is it a good idea to include the discount factor in grid-search on meta-parameters of your algorithm? If yes, explain why; if not, would you still consider a range of discount parameter values and why?

   Answer:

   The discount factor is part of the environment. In the chess game, the reward is only dependent on whether we won or lost. The rule of repeating a position three times means that the game has a finite horizon. Hence, the discount

factor should be one. Grid-search on the value of discount factor might sometimes give good results for γ close to zero. This, however, in the case of chess, indicates either an ill-working algorithm or too short learning (i.e., too few learning episodes) as it is easier to learn when you limit the horizon. In an extreme case, when $\gamma = 0$, no learning is needed at all. In the case of chess, we might still consider values close to one but not one exactly (i.e., 0.99 or 0.999) as it improves the numerical stability of some algorithms by distinguishing paths to terminal states of different lengths.

In the case of finite-horizon game, when winning/loose situations are only necessary, it can be mathematically proven (try it yourself) that value of a state is equal to the expected number of wins weighted by the terminal state rewards.

There are still some scenarios when grid search on discount factor is needed, especially when the discount factor is unknown and cannot be estimated. However, caution is needed.

6. (2 points) Suppose that

$$\max_{x \in X} |r(x)| = r_{\mathsf{max}}.$$

Using only r_{max} and γ, give the tightest lower and upper bound on the cumulative discounted reward in a single episode.

Answer:

The maximum is equal to

$$r_{\mathsf{max}} + \gamma \cdot r_{\mathsf{max}} + \gamma^2 \cdot r_{\mathsf{max}} + \cdots = \frac{1}{1-\gamma} r_{\mathsf{max}}.$$

The minimum is the same except with the minus sign.