DEEP REINFORCEMENT LEARNING 2023

Homework 01

Author
Krishnendu Bose, Jakob Heller, Alexander Wallenstein
Group 02
Osnabrueck University

27.04.2023

1 Task 01

You are tasked with creating an AI for the game of chess. To solve the problem using Reinforcement Learning, you have to frame the game of chess as a Markov Decision Process (MDP). Describe both the game of chess formally as a MDP, also formalize the respective policy.

1.1 Solution

Definition 1.1 *Markov Decision Process (MDP)*
A finite Markov Decision Process consists of:

- the finite set of States S, with states $s \in S$,
- the finite set of Actions A, with actions $a \in A$,
- the probabilistic state dynamics $p(S_{t+1}|S_t, A_t)$,
- the probabilistic reward dynamics $r(s, a) = \mathbb{E}[R_{t+1}|s, a]$

- **States** S are the possible configurations of the chess board. Although really large (some estimations for *legal* chess positions are around 10^{45}), that number is indeed finite.
- **Actions** A are the possible moves a player can make. This includes all possible moves for all pieces, including castling, capturing, promoting and en passant.
- **State Dynamics** $p(S_{t+1}|S_t, A_t)$ is the probability of the next state S_{t+1} given the current state S_t and the action A_t. Note: given a state S_t and an action A_t of our player, the next state S_{t+1} is the board state after the opposing player has made their move. p is therefore not deterministic, and includes every possible move the opposing player can make after A_t.
- **Reward Dynamics** $r(s, a) = \mathbb{E}[R_{t+1}|s, a]$ is the expected reward after taking action a in state s. The reward is 1 if the game is won, 0 if the game is drawn, and -1 if the game is lost. We actively decide against introducing a reward for capturing pieces or similar subgoals: otherwise, the agent might learn to play for the reward instead of playing to win. "[...] the reward signal is not the place to impart to the agent prior knowledge about *how* to achieve what we want it to do." [1]

Our policy π, or $\pi(a|s)$, is a probability distribution over actions a given a state s. A complete policy chooses a move (optimally the best move) for every possible board state. In chess, a good policy has very large probabilities for moves that lead to a win, and very small probabilities for moves that lead to a loss.

2 Task 02

Check out the LunarLander environment on OpenAI Gym. Describe the environment as a MDP, include a description of how the policy is formalized.

2.1 Solution

There are multiple variations to the Lunar Lander environment —because nothing was specified, we will choose the discrete version without wind.

- Different to the chess example, we only have an observation space, not a state space. The observation space consists of states s, where each state is an 8-dimensional vector. Each state vector contains x and y coordinates of the lander, its x and y linear velocity, its angle, its angular velocity and two boolean values indicating whether the left and right legs are touching the ground.
- The set of **actions** A is discrete and includes the following actions: do nothing, fire left orientation engine, fire main engine, fire right orientation engine.
- The **state dynamics** $p(S_{t+1}|S_t, A_t)$ are deterministic. The next state S_{t+1} is determined by the current state S_t and the action A_t.
- A **reward** is assigned to every action, the reward of the episode is the sum of all rewards. For each step, the reward:
  - is increased/decreased the closer/further the lander is to the landing pad.
  - is increased/decreased the slower/faster the lander is moving.
  - is decreased the more the lander is tilted.
  - is -0.3 for each step where the main engine is firing,
  - is -0.03 for each step where the left or right orientation engine is firing,
  - is $+10$ for each leg that is touching the ground,
  - is -100 if the lander crashes,
  - is $+100$ if the lander is on the landing pad,

  If it scores at least 200 points, the episode is considered a solution.

Since we are dealing with an observation space, we cannot use a policy $\pi(a|s)$ that maps every state to an action. Instead, we use a policy $\pi(a|o)$ that maps every observation to an action. The policy is a probability distribution over actions a given an observation o.

3 Task 03

Discuss the Policy Evaluation and Policy Iteration algorithms from the lecture. They explicitly make use of the environment dynamics ($p(s', r|s, a)$).

- Explain what the environment dynamics (i.e. reward funxtion and state transition function) are and give at least two examples.
- Discuss: Are the environment dynamics generally known and can practically be used to solve a problem with RL?

3.1 Solution

4 References

[1] R. S. Sutton, F. Bach, and A. G. Barto, *Reinforcement learning: An introduction.* MIT Press Ltd, 2018.