Homework 1

Task 1

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.

The game of chess is a fully observable deterministic zero sum game that can be fomulized as an MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$. Its states $\mathcal{S}$ are those that could be reached in a game via legal moves. Its actions can be formulized as the subset of the cartesian product of board positions initially ocupied by the moving figure B, figures F and board positions that the figure moves to $B \times F \times B$. From this product a subset $\mathcal{A} \subset B \times F \times A$ is constructed, where each tuple $\langle b_{init}, f, b_{final} \rangle$ satisfies the *rules of movement for a figure*. Castling is conceptualized as a move with $f = \mathbf{King}$. To further get the subset of legal moves given a state $\mathcal{A}(s) \in \mathcal{A}$ the conditions of the *games rules* are applied with respect to the *current state*. The transition dynamics $\mathcal{P} \subset \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ are specified for all legal moves $\mathcal{A}(s)$ in s and equate to 1 if s' is the subsequent state and 0 otherwise. The state resulting from a performed in s will be abreviated to $s' = T(s, a)$. The reward function could be set to $r(s) = 1$ if the agent won the game in s, $r(s) = -1$ if the agent lost the game in s and $r(s) = 0$ if the game terminated in a draw or is still ongoing.

To capture the MDP as a single agent process even though two players are needed, the agents opponent can be integrated in the MDPs transition dynamics, such that the state resulting from an agents action is the state that is reached after the agents action is performed first and the opponents thereafter.

The optimal policy satisfies $\pi^*(s) = \arg\max_a v^*(T(s', \arg\min_{a'} v^*(T(s', a'))$ with $s' = T(s, a)$.

Task 2

For the LunarLander environment, the corresponding MDP can be formunlized as $\mathcal{S} \subset \mathbb{R}^2$ where not reachable coordinates are excluded, $\mathcal{A} = \{\text{noop}, \text{left}, \text{up}, \text{right}\}$, $\mathcal{P}$ implements the environments physics, terminating the episode if the Lander crashes or comes to rest and $\mathcal{R}$ rewards states as follows:

- $+10$ additional for each leg on the ground

- -0.3 for firing the main engine
- $+100$ for landing on the landing pad
- -100 for landing outsinde the landing pad
- $[+100, +140]$ for moving towards the landing pad

Task 3

The Policy Evaluation algorithm takes a policy and applies the bellman operator to a randomly asigned value function over all states until the maximal error has fallen under a specified threshold. The Policy Improvement algorithm takes a value function and applies the bellman operator for optimality on a policy over all states. The Policy Iteration algorithm alternates between the two algorithms to achieve a better estimation of the policy at hand and than use that estimation to improve the policy. All of the those algorithms are dependend on knowing the transition dynamics.

- The transition dynamics are the underlying matrix of the Markov Chain that determines the MDPs. Those dynamics determine the probability of the outcome of action given a state in which the action is performed. The reward dynamics specify the notion of optimality that the agent should reach in the problem that is modelled by the MDP. One policy that optimizes for one reward function in a MDP can also be optimal for another reward function in that MDP. For example: An agent is tasked with controlling a fusion reactor. Its transition dynamics are simply the physical laws of the universe and it has one really straight forward reward function $r(s) = \Delta E$, maximizing the cummultative energy output over time. A slightly modified reward function could additionally penelize the agent if it sets the controlls to parameters that are known to be unstable and might potentially break the reactor. The optimal policy might however be identical since the drop in future reward after the reactor has been damaged as a result of setting the controlls to dangerous parameters is implicitly coded into the environment dynamics. The difference lies not in the resulting optimal policy, but in the path that the policy takes towards its goal during the training.
- In general, transition dynamics are not known by the agent but are in fact knowable and learnable from experiance. In many domains (mainly those that do not rely on major planning) sampling experiance from the environment gives a good estimate that can substitute the explicit callculation of the expectation.