DRL Homework 01

1.1

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 set of states S:

* The state space: S = {s1, s2, ..., sN}, where each state si corresponds to a unique configuration of the chess board
* 8x8 field, each field can have 6 different game figures in two different color
* 64^6^2 🡪 64^36 🡪(2^6)^36 🡪 2^(6\*36) 🡪 2^216
* This is still very large. One could reduce this further since not all possible states are legal states in the game

The set of Actions A:

* The action space: A = {a1, a2, ..., aM}, where each action ai corresponds to a legal move that a player can make in the current state
* The moves a Player can make
  + Moving a pawn one or two squares forward on its first move and one square forward on subsequent moves, except for capturing an opponent's piece.
  + Capturing an opponent's piece with a pawn by moving the pawn one square diagonally forward to a square occupied by an opponent's piece on an adjacent file.
  + Moving the king one square in any direction, as long as the move does not result in the king being in check.
  + Castling, which is a move where the king is moved two squares towards a rook, and that rook is moved to the square over which the king crossed, as long as neither the king nor the rook has moved previously, the squares between the king and the rook are unoccupied, and the king is not in check.
  + Moving a knight to one of eight possible squares in an L-shape pattern. Knights are allowed to "jump" over other pieces.
  + Moving a bishop diagonally any number of squares, as long as there are no pieces in the way on the diagonal and the move does not result in the king being in check.
  + Moving a rook horizontally or vertically any number of squares, as long as there are no pieces in the way on the file or rank and the move does not result in the king being in check.
  + Moving the queen in any direction along a straight line, either horizontally, vertically, or diagonally, as long as there are no pieces in the way on the path of the queen and the move does not result in the king being in check.
  + Capturing an opponent's piece by moving a piece to the square occupied by the opponent's piece.
  + En passant, which is a move where a pawn can capture an opponent's pawn immediately after it moves two squares from its starting position and lands next to the capturing pawn on the fifth rank.
  + Promotion, which is a move where a pawn can be promoted to a queen, rook, bishop, or knight when it reaches the opponent's end of the board.
  + Moving the king out of check by moving it to a square where it is not under attack by an opponent's piece, or by capturing the attacking piece.
  + Blocking a check by moving a piece in between the attacking piece and the king, as long as there are no pieces in the way on the path of the blocking piece.
  + Capturing a checking piece to get out of check.

The transition function:

* The state transition function: T(s, a, s') = P(s' | s, a), where T(s, a, s') is the probability of transitioning from state s to state s' after taking action a, given that the current state is s
* The state transition function would be a complex function that takes into account all the possible moves that can be made from the current state, and the resulting board configurations that would arise from each of these moves, as well as the probabilities of those outcomes.

The reward function:

* The simplest reward function would be +1 for a win, -1 for a loss, 0 for a draw
* However, a more complex reward function would be possible. E.g. positive reward for capturing an opponents piece, checking the opponent, negative rewards for loosing a piece
* Even more sophisticated reward functions would analyze the resulting state in regards of general pressure, discovered attacks

The policy:

* π**(a|s)** to be read as: the probability of taking action a ∈ **A** when encountering state s ∈ **S**
* In the context of chess, the policy would provide a way for an AI agent to select the best move to make at each turn, based on the current state of the board.

1.2

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

* Set of states S: The state space S of the Lunar Lander environment is a vector of the lander's position, velocity, angle, and angular velocity and two booleans whether each leg in on the ground or not.
* Set of actions A: The action space A of the Lunar Lander environment includes do nothing, fire main engine, fire left engine, and fire right engine.
* Transition function: The state transition function P(s, a, s') is the probability of transitioning from state s to state s' when taking action a.
* Reward function: The reward is a combination of several factors, including the lander's position, velocity, angle, number of legs in contact with the ground, and engine firing. An episode is considered a success if the lander lands safely, which grants a reward of +100, while a crash results in a penalty of -100 points.
* Policy: The policy π(s) is obtained by selecting the action with the highest expected sum of rewards given the current state s, represented as π(s) = argmax\_a Q(s, a).

1.3

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 function and state transition function) are and give at least two examples.

The environment dynamics in a reinforcement learning problem refer to the rules governing how the agent's actions affect the state of the environment and the reward that the agent receives. Two examples of environment dynamics are:

* In the game of Pac-Man, the state transition function would be how the position of the Pac-Man and the ghosts change after each move. The reward function would be how much reward the agent receives for collecting pellets, eating fruit, and surviving for longer periods of time.
* In the problem of training a self-driving car, the state transition function would be how the car's position and speed change after each action, such as accelerating or turning. The reward function would be how much reward the agent receives for staying in the center of the lane, obeying traffic signals, and avoiding collisions.

• Discuss: Are the environment dynamics generally known and can practically be used to solve a problem with RL?

* The environment dynamics are not always known beforehand and can be difficult to model. Especially in complex real-world scenarios. However, in many cases, the dynamics can be estimated and fine-tuned. Thus RL can be applied in a wide range of problems, from playing complex games to controlling robots and autonomous cars.

2.1

Look up some examples of GridWorld! List at least three links or references to different GridWorlds you could find online.

“frozen lake”

[https://gymnasium.farama.org/environments/toy\_text/frozen\_lake/#frozen-lake](https://gymnasium.farama.org/environments/toy_text/frozen_lake/" \l "frozen-lake)

“cliff walking”

<https://gymnasium.farama.org/environments/toy_text/cliff_walking/>

“taxi”

https://gymnasium.farama.org/environments/toy\_text/taxi/