hugomcp@di.ubi.pt, 2022/23

Artificial Intelligence

Practical Project 1 – Chess Intelligent Agent

As you might know, Chess is a board game for two players. It is sometimes called Western chess or international chess to distinguish it from related games such as xiangqi and shogi. The current form of the game emerged in Spain and the rest of Southern Europe during the second half of the 15th century after evolving from chaturanga, a similar but much older game of Indian origin. Today, chess is one of the world's most popular games, played by millions of people worldwide.

Chess is an abstract strategy game and involves no hidden information. It is played on a square chessboard with 64 squares arranged in an eight-by-eight grid. At the start, each player (one controlling the white pieces, the other controlling the black pieces) controls sixteen pieces: one king, one queen, two rooks, two bishops, two knights, and eight pawns. The object of the game is to checkmate the opponent's king, whereby the king is under immediate attack (in "check") and there is no way for it to escape. There are also several ways a game can end in a draw.

The first practical project regards the implementation of an “Intelligent Agent”, in Python language, able to play Chess against an opponent, in a **server-client framework**.

As illustrated below, the framework is composed of a “Server” (supplied by the Teacher), where two “Client” applications should connect to. The idea is that each student is responsible to implement its own “Client” (according to the skeleton provided in the course web page).

The server-client framework, written in Python, is available at the course web page.

It is expected that each client:

1. Keeps an internal representation of the game state;
2. Receives information about the current game state;
3. Decides its own move, to maximize the chances of winning the game, and returns the updated state of the game.

Game States

Each game state will be stored in a string, of length 64, using the following convention:

a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p:
(a: Black Tower, b: Black Horse, c: Black Bishop, d: Black Queen, …i: Black pawn,…p: Black pawn)

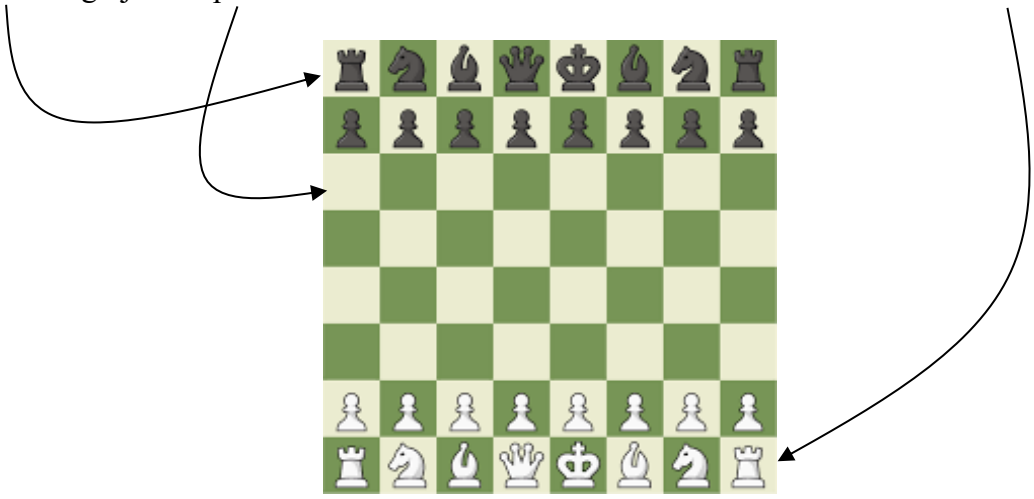
A, B, C, D, E, F, G, H, I, J, K, L, M, N, O , P:
(A: White Tower, B: White Horse, C: White Bishop, D: White Queen, …I: White pawn,…P: White pawn)

z: Blank position

This representation starts from the upper left corner, representing the black pieces side of the board, and goes in right/down directions.

Example – Initial state of the game:

“abcdefghijklmnopzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzIJKLMNOPABCDEFGH”



- Full information about the Chess rules can be found at: https://en.wikipedia.org/wiki/Rules_of_chess

- The unique changes to the classical Chess rules is that “Castling” and “En passant” rules will not be considered.

- The intelligent agent must guarantee the validity of its own move.

- The first player to return an invalid move loses the game immediately.

- White pieces always start.

- If there are N (server-side parameter configuration) moves without a capture, a draw will be declared.

Evaluation

- The goal is to develop the Checkers agents during the practical classes of the course, where each student should implement the most complete/smart heuristics as possible.
- The evaluation of the works will be comprised of two parts:

    - **Written report (70%)**, where the student summarizes his most important choices taken during the development of the work.

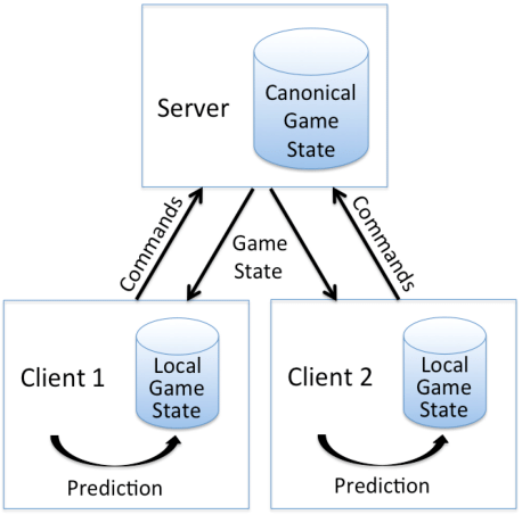
    - **Performance (30%)**. Each intelligent agent will play against the agents developed by all other students, earning 2 points for each victory, 1 point per draw, 0 points for each defeat, and -1 for defeats with invalid moves.
        - At the end, the points accumulated by each student will be summed up and a table will be produced, ranking the agents from the best to worst.
        - The top performing student will earn the full 30% of the mark, while the remaining ones will earn in direct correspondence to their position in the rank.
        - Example. Having 5 students (A, B, C, D, E), ranked from the best (A) to the worst (E) performance, the corresponding marks will be:

          A – 30%
          B – 22.5%
          C – 15%
          D – 7.5%
          E – 0%

Python Framework

This work should be developed according to a typical "server/client" architecture, illustrated in the Figure below. The server will be responsible for controlling the game moves and check their validity. Each client is responsible for – upon the reception of a game state – return the updated state, corresponding to the movement of the player.

The server will write a log file, describing all movements taken during the game, along with information about the winner.



To start a new game, just have to…

1. Start the server:

   python3 server.py <host> <port>

   Example: start a server at the "localhost", and listening at port 5555

   python3 server.py 127.0.0.1 5555

2. Start the (two) clients, using the following protocol

   python3 client.py <host> <port> <nickname> <player>

   where "nickname" identifies the player and <player> is a binary flag (0=White | 1=Black), determining the used pieces.

   Example:

   python3 client.py 127.0.0.1 5555 hugo 0
   python3 client.py 127.0.0.1 5555 pedro 1