Siddharth Parasnis / AIGames: FreeMaxB / GitLab: sparasnis

Dr. Patrick Taylor

Intro to Artificial Intelligence

5 May 2016

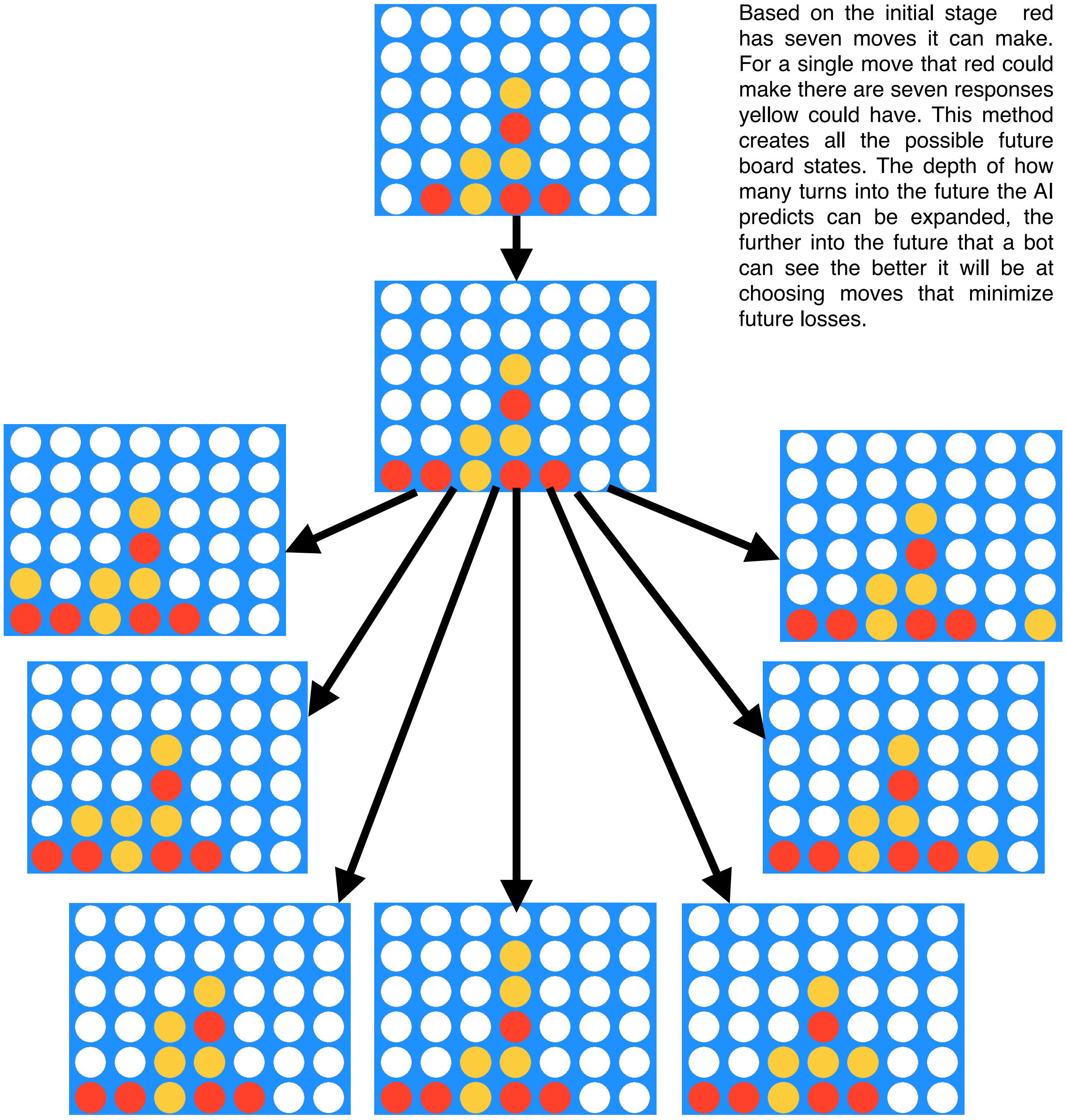
Theory of a Connect Four Bot

Connect Four like many board games can be played using AI agents. In Connect Four, players take their turn alternatively, each one placing one of their own discs in one of seven columns to attempt to have four of their pieces in a straight line. Connect Four is a game with perfect information, each player has a complete information about what is the current state of the board is. Each player also always has all the information on what possible moves and states of the game can happen next. It is also a zero sum game because each player’s advantage when they drop a piece in becomes another player's disadvantage. As a result, a good AI bot has the information that is required to predict which states will be optimal for the bot. The bot will make decisions based on the best outcome that it predicts can happen, and the best outcome for the agent is often minimizing the losses made by the opponent. This is the basic idea behind using a minimax algorithm to play Connect Four.

The Connect Four board is given to the agent as a string of comma separated value with a list of what is in each slot of the board, a “0” represents an empty spot in the board, a “1” represents where our bot has placed a disc, and a “2” represents where the opponent has placed their chip. Each time it is an agent's move, the server gives the agent the state of the board and the agent responds with a move. A move is represented by the integers zero through six to represent which of the seven columns the user is placing the disc into. There are three object classes in the connect four bot. The Field is an object representation of the board which has various methods to check if a move is valid and what the state of the board is. The BotParser is what interacts with the server, it saves the settings of the game, reads what is the state of the game board, and requests moves when they re required. The BotStarter class has the method makeTurn which sends a move back to the server. makeTurn is where the “intelligence” of the bot is used, it calls on helper functions to rate each column to choose a move.

A casual human connect four player would probably play Connect Four in a different manner than an artificially intelligent bot. A simple strategy for a human connect four player could be to try to build certain patterns that they know could eventually form connect fours in the future and to put discs in columns to try to prevent an opponent from getting connect fours. This means that it is more likely to place discs in columns that are near other columns. A naive heuristic could be to give a higher priority to moves which are closer to spaces on the board that contain other discs. However, a random placement of the discs with a higher priority on columns which are closer to where the most discs are does not work out well. It is clear that randomly selecting columns is not the most strategic way for a bot to operate.

The advantage which the AI bot has over a human AI is the ability to quickly process many future states which the board can hold. In fact the number of possible board configurations in a seven column wide, six row tall, connect four board is 4,531,985,219,092. This is a huge space but the A bot still only needs to examine a fraction of this space. There are seven possible moves that the AI can make, so a simple AI can examine all the seven moves and if any of them create a connect four then that is an option then that move should be played. But if there are no opportunities to make a connect four then how does the bot know where to make a move? The solution is to keep examining future game states. In the function oneMoves, an array of Field objects, or even a single Field is passed. For each of these Fields, seven new Fields are created, each with the bot’s own disc placed in a different of the seven columns. This represents all the moves a bot can make given the state of a game. It then returns this new array of Fields that is now seven times longer. The function twoMoves



Based on the initial stage red has seven moves it can make. For a single move that red could make there are seven responses yellow could have. This method creates all the possible future board states. The depth of how many turns into the future the AI predicts can be expanded, the further into the future that a bot can see the better it will be at choosing moves that minimize future losses.

is a similar function except that it places a two in each of the columns in each of the fields. By passing the current state of the board to oneMoves and then passing the output to two moves, and then passing the output back and forth, it creates a tree of possible fields which are all the possible next moves of the game for both players turn. Each time the functions which add moves and expand the tree are used, they also evaluate what sort of opportunities are created when a disc is added. Evaluating only the move which was just played saved computation time because the bot does not have to analyze every piece on the board, just the move which was played, similar to how a person The level of depth is controlled with the function moveWithDepth which computes the future states at a depth of the integer that is passed to it. A depth of one would be one turn of the bot moving and one turn of the opponent moving. If adding the disc for the bot creates two, three or four in a row near each other, then this increases the rating of that move. If adding a disc for an opponent creates two, three, or four within the space around the added disc it lowers the rating of that move. The rating of each move is stored in an array of length seven. A four in a row would affect the rating much more than two within the space of four because two within the space of four is common but four in a row could win or end the game.

Overall, using a minimax algorithm to examine the future possible board states and choose moves which minimize loss and increase the chances of getting a connect four is a good way for a bot to play connect four. The bot will always be analyzing both it's own moves and the moves of the opponent to plan for the future. The variability in the bot can come from the level of depth which it checks into the future, which is how many moves ahead it considers. The further into the future the bot can see, the better it will perform. The bot must strike a balance between a large enough depth and it's limit on time to respond with a move. Each level of depth increases the space in 49 times it's size because it computes seven possibilities for the bot for a board state, and seven possibilities for the opponent. Connect Four is a solved game, meaning that the first player has a winning strategy which if followed guarantees a win. The time limit on moves however, means that it is very hard to compute the winning strategy and allows for AI techniques to be used instead to play the game.