Connect Four and Machine Learning

COMP 452 – AI for Game Developers

April 30, 2024

Keith Mitchell

SID 3178513

Connect Four

Game Logic and Design

Game Logic

The game logic for connect four is predefined so not much flexibility in how the game plays. I chose to go with red and yellow instead of black and white, as those were the colours for the versions that I played growing up. Following the rules, the player goes first and is the red colour. The AI takes the yellow colour.

Game Engine

Nothing new to note here. The only major change I can think is that I stepped away from using the Tile object I have used in the past and instead opted to represent the board with a simple enum. One of the main reasons for this was to speed up the minimax solving so it only has to copy an enum array instead of an array of tile objects. Otherwise, the logic of the engine is really the same core structure I have used for the past projects.

Compiling/Game Setup

Before you can run the visualizer you must first ensure that the Java Runtime Environment (JRE) is installed on your system. If you are wanting to compile the program yourself, you will also need the Java Development Kit (JDK) installed.

Compiling:
In order to replicate the development environment, I used the IDE IntelliJ. You can load the ConnectFour folder as a project. Depending on your compiling environment you may need to target the assets folder as the resource folder. You can usually do this by right clicking the folder in the IDE and marking it as the resources folder, though each IDE may differ. You then can then use the IDE to compile/build/run the game. In IntelliJ, open the ConnectFour file (in the game package), and click play (Shift+F10).

You can compile without an IDE. Generally, this will involve navigating to the project folders and using command line prompts to compile all the .java files using the command line javac. This turns the files into .class files which can then be run using the java command line.
(ConnectFour.java is the main entry point file).

If for some reason the assets cannot be found and you are getting null references, this might be due to miss targeting of the assets folder. The only location needed to adjust will be within the ResourceLoader class. Adjust the folder directory to match your systems directory structure to the location of the assets folder.

Running from .jar:
The .jar file is located in the ConnectFour folder. On most systems you can simply double click the .jar file and the game will launch. (Assuming you have installed the JRE).

To run the game using command line you can navigate to the folder containing the .jar file and the enter the command: java -jar ConnectFour.jar

Minimax Algorithm

I had a ton of issues with this alogirthm, primarily because I first tried to score the board from the perspective of the player that made the move. This should work but for some reason I was never able to properly flip the scoring even though I was following negamax. Something was causing a hitch. The moment I simplified to just always scoring the board from the computers position and then minimizing when the player and maximizing when the ai, it worked. I still cannot explain why my original solution wasn't working as I liked the idea of scoring from the viewpoint of the individual that made the move.

For the board scoring I figured I would try and just score every possible grouping of 4. After doing some research into connect four solves, this appears to be a strong approach though I did see variations that included valuing the middle, scoring blocks, and some that tried to assess setting up multiple angles. I chose to start with just assessing all groupings of 4 and came to some realizations through testing. In order to check values, I tried to see how decisions work at different depths. Interestingly, even with a depth of only one the AI is incredibly efficient though it will fail to block 3 in a row since that placement only removes a 3 in a row score from the player. If it has a move that will maximize its score not realizing that if it doesn't go, there it will lose. This is obviously why a depth of at least 2 is necessary otherwise blocking an instant win will go neglected. The main reason for this is because of the way the board is scored. The route I went with doesn't complicate its scoring by scoring for blocks, or the strength of the middle position. The reason for this is that it kind of does both naturally. By blocking a 3 in a row position, the opposing players score that they were getting from a potential 4, is removed. The block does not aid the player that placed it, it simply removes the scoring potential that the other player had. It may also aid in setting up rows, and those would be calculated, but adding extra weight to block just was unnecessary. In regard to the importance of the middle row, this naturally gets assessed as EVERY set of 4 positions is calculated. That means the middle row is naturally included in the most checks. If you place a token on the edge the computer will take the middle slot. This is because it has the most scoring potential. By scoring the potential strength of the set of 4 and removing score for the players sets made checking board scores really easy. For the actual values and weights applied to 1, 2, and 3 in a row, I messed around with values and asked ChatGPT what I should set them to and the values I settled on seem to work well. Technically the set of 4 should never be scored as it would be considered a game over board and I monitor this separately. I left the 4 check in from earlier builds.

When solving for game over I simply just check the new position to see if that token creates a win. Instead of checking the entire board for a win, I can make very few checks since I already know the board prior was not a game overboard, otherwise the game would have ended. Since I know what

the new move is I can just check that move. Then when I score the board, I set to max or min value first if it was a winning board to avoid evaluating the full board.

How To Play

Move the mouse around and click to place your token. The green token represents where your token will fall. After clicking the AI will place their token and play returns to the player. Continue back and forth until a winner is found. If a winner is a found the game will end and the winner is awarded a point. Click the checkmark button to start the next round. A tally of total wins is in the top left with the player score on the left and AI on the right. (Home vs away logic from sports, player being the home slot). In the top right corner is a forfeit button which the player can click to forfeit the round and start the next one. This will count as a loss, but the next round will start right away. If a tie occurs, then neither player gains a point.

Bugs/Future Builds

Bugs

This was incredibly challenging to bug check since checking to see if the AI is working correctly and checking a 4 deep board placement for accuracy is incredibly challenging. I did run some checks to see what the output was for score and a visual of the board and it seemed to be solid choices. I also played many, many rounds against the AI and was only able to beat it once or twice. Other then the AI logic, I do not know of any game crashes or bugs. In theory if the game ran long enough the win and loss tracker would bug out but its incredibly unlikely to experience that.

Sound/Animation Systems

There is still no sound system built and though sprites exist, I did not add any animations to this game. I debated having the tokens spin down from the top as if being dropped. If I were to spend more time with this that would definitely be the next step to liven up the game.

Functionality

There is a lot of extras I could add to this. A splash menu with a play button, sound, animations. I did add a score tally and took some extra time to track a draw line for the winning set of 4. This helped visualize the win and make it feel more like a game. I would also add a timer to slow down the Ais turn to make it feel like its thinking. Having a 2 second delay with a “Computers Turn…” message, would make it feel more realistic.

Lastly, I could improve the algorithm by adding alpha and beta pruning and making other improvements. For one, I could keep the created board states that are children/future moves off the move that is chosen. Then when the new turn needs to be assessed, I would first check which of 7 positions were chosen by the player and move down to that board. From there I would already have the next 2 rows built (assuming 4 depth), and would just need to build the next two rows after. This saves the processing time of copying over and building the same boards again. There is a bunch of

complexity that this would add as it would require closer tracking of board references/building a tree structure.

Overall Thoughts

I had to do a massive refactor on this project after having it working fine. First, I started out by using Tile object class which was a massive headache when it came to copying and duplicating the board. Second, I had the game working when it was only picking the best of 7 options (1 layer deep), but I completely flopped on my first few attempts at recursively selecting the correct board on each layer (player vs ai turns). I just couldn't figure out what was wrong and finally just reset the way I was trying to do it. Happy with how it turned out though.

Machine Learning

Game Logic and Design

Game Logic

I was really struggling to come up with a build for this one but finally settled on the idea of changing the Ant simulation to use Q-Learning. The ant colony would has a hive mind that learns the layout and maps the region. Every ant adds to the overall knowledge of the hive. This would be a neat way to visualize the colony slowly getting better and more efficient. Once a food source was found the ants would start to take the same paths as it becomes the optimal route creating the real life visual of a trail of ants moving towards the same food source.

Game Engine

Build a class that holds the Q-Learning logic. Would have 3 Q-Learning boards, one for each goal. Find food, head home, find water. This separation of goals would allow for a clearer Q maps.

Q-Learning

Unfortunately, this build was one I really struggled to execute on. I think with another day or two I could have worked my head through it but I was really struggling to grasp how to get started with this. From what I understand, I needed a way to represent the state of the board. This was a huge issue as I couldn't figure out if this needed to include the entire board or if it should only include the neighbouring tile. Since the state of the board never changes would it matter? After this the ants would need to know that they can move up, left, right and down (and wall collision). Then hitting a food tile would reward the while finding poison would have a negative effect. Then as the ants moves around they update the Q values as they explore the space. The longer it runs for the more accurate the data will get. If an ant finds food it just switches its goal and the Q Table to match and repeats the process with the reward being for finding the home tile.

One issue that I kept hitting with this is I don't understand how the table would update towards finding the food. Wouldn't the only tile that sees the reward be the final movement onto the food tile. That specific movement would be rewarded. But the movement to the tile before that wouldn't know that the path it took was a good one. So how would the path populate forward to the reward. Would the only way to solve this be knowing the total distance to the food and reward getting closer to it? Wouldn't this defeat the purpose of "finding" the food? I kept getting stuck in this loop and I couldn't figure out the solution and was unable to plan the code as a result.

I wish I had properly figured out a way to incorporate this as I think it would have been a neat way to visualize the Q-Learning process, especially if I showed the values on the board like I did with the pathfinding visualizer. I may still hammer away at this in my spare time but unfortunately, I ran out of time, even with the extra 3 days.