Aughdon Breslin and Jason Teets

Learning Module 8 - Connect Four AI

Initial Proposal

Our goal in this learning module is to create an AI that can play connect 4 using the minimax algorithm. Minimax is an algorithm that allows the computer to look ahead several moves along potential branches using recursion. We will also use a technique called alpha beta pruning to improve the efficiency of our algorithm so the AI can look ahead several moves more than it would have been able to without freezing up.

We will probably code the connect four game itself in pygame, perhaps using sublime text editor to code it. After coding the game in two or three days we will begin implementing minimax. We will spend a few days setting that up and fine tuning how position strength is scored. Position strength is essentially how the computer determines how good an orientation is. It will incorporate the presence of a winning move, three in a rows, pieces in the center and any other factors we determine to be integral to winning. The more factors we include and the more fine tuned the impact of those factors on board score, the better that board. In terms of learning minimax we will probably use pages describing the algorithm and if we get stuck a tutorial or two. We will do the same for alpha beta pruning. Alpha Beta pruning essentially realizes at a certain point along a branch of moves that that branch is not worth checking any further.

After this part is complete we plan to integrate our project with Rob and Liz’s physical boards. Possibly using network tables we will have our game communicate to arduino and display the board state from the computer on the physical board. The best case scenario would be the computer being able to play and beat competent players on a physical board.

We will both be working on researching and coding the game as well as implementing the algorithms. As of yet we don’t foresee a divergence of roles. For Days 1-5, we will build the pygame connect 4. Days 6-9 we will develop the user interface. From Days 10-13 we will try to implement and fine tune our algorithms for AI. Days 13-18 we will integrate our AI with Rob and Liz’s Connect Four LED board. Then, Days 19-22 we will interface the game with Liz and Rob’s physical board.

1. Day 1 Log (4/16/19): We started off planning what is needed to be done for the Connect Four board operations as well as how the computer’s side will work. We designed our connect four board, added in move mechanics, as well as organized the code.
   1. (4/16/19) In study hall, Jason developed the check horizontal and vertical conditions and tested them. He also flushed out some other methods. (8:40-9:40)
   2. (4/16/19) At home, Aughdon developed the positive and negative diagonal win conditions for the Connect Four board, and added in some more friendly user interface code (3:00pm - 5:00pm).
2. Day 2 Log (4/18/19): We focused on the computer’s strategy today. We looked into tutorials for how a computer would check all possible outcomes and how it would weigh winning conditions.
   1. <https://www.youtube.com/watch?v=MMLtza3CZFM> 1:15:20
   2. (4/20/19) At home, Jason developed the find three and find two methods for scoring the board for the AI to process as well as the find center method. (10 am-12 pm)
3. Day 3 Log (4/25/19): Today was spent debugging the Computer’s turn. We had to rework the conditions a lot and define new terms within to try and make the recursive nature of checking for the best move possible. We still are running into errors with recursion and data types.
4. Day 4 Log (4/26/19): Jason is at PMC today. Audie worked on developing the scoreboard that decides what move is better than another. He also worked on debugging the AI’s evaluation of the board based on these scores.
5. Day 5 Log (4/30/19): We tried various methods to fix a resilient bug that refuses to be solved. We could not accomplish much today because the logic within this problematic method was reviewed multiple times and yet the bug prevailed.
6. Day 6 Log (5/1/19): We spent a lot more time debugging, but we were able to find one of the bugs and found out we were still running into logical errors. We decided to pseudocode the function entirely again and recreate the program based on this without looking at the first version in hopes to somehow beat the bug.
7. Day 7 Log (5/2/19): We rewrote the miniMax program hoping to no longer encounter the bug. We ran into different errors however, which is really good. We’ve stopped encountering the None type errors or recursion errors but some logical errors are still appearing. The board is filled on the first turn despite all the moves being made on a copy board.
   1. Jason at home (6:45-7:45 pm 5/5/19): I changed how we copied the array to get our connect 4 game to work.
8. Day 8 Log (5/6/19): Jason was not here today for AP testing, and Audie tested the game at lower-level depths and looked into how Jason and he could integrate their Colab code with the Arduino. We ironed out the miniMax program to remove the rest of the bugs and started to play actual games against the AI we created.
9. Day 9 Log(5/7/19): Audi and Jason tested the game and implemented alpha beta pruning. It now runs more efficiently and can reach higher depths. Alpha beta pruning was actually pretty simple to implement, and just removes some possibilities that would be considered bad moves so the computer doesn’t have to run through them.
   1. <http://www.toptechboy.com/arduino/python-with-arduino-lesson-2-installing-the-software/>
10. Day 10 Log (5/8/19): Both Audie and Jason are out today for AP testing.
11. Day 11 Log (5/10/19): Both Audie and Jason are out today for AP testing.
12. Day 12 Log (5/13/19): Jason is out today for AP testing, Audie worked on GirlsGo CyberStart
13. Day 13 Log (5/14/19): Jason and Audie are out today for AP testing.
14. Day 14 Log (5/16/19): Jason is out today for AP testing. Today Audie brainstormed how to integrate our Colab code into Liz and Robs Arduino LED board. We decided we could throw our code onto a Pi, and send a coordinate through bluetooth to Rob & Liz’s Arduino and they’ll light up the board from those array coordinates.
15. Day 15 Log (5/17/19): We began downloading software to allow us to communicate between our python code and the arduino. We are trying to avoid re-coding the entire thing in C++.
    1. Jason Hours 5/19/19 (9-12 am): I went through code academy’s C++ course to familiarize myself with the basics of the language in case recoding our python code became the best option.
16. Day 16 Log (5/20/19): It was a shortened block today due to NJSLA testing, so we brainstormed how we could go about making our Python code readable to an Arduino. We may have to manually translate our code to Colab, probably doing about 60 lines a day, or we could possibly use Xander and Sal’s project to transfer information from a Raspberry Pi to a Bluetooth module on an Arduino.
    1. At home (8-10pm), Aughdon researched possible techniques for Arduinos to read Python code, but they seemed really convoluted, and the guides weren’t very clear. Our best option seems to be C++ translating, which will be time-consuming to say the least.
17. Day 17 Log (5/22/19): Both Jason and Audie were out today for WHAM.
18. Day 18 Log (5/23/19): Audie was out today for AP testing. Jason converted more lines from Python to C++.
19. Day 19 Log (5/24/19): We converted a lot of lines to C++ today and started debugging the C++ errors. Translating Python to C++ directly is pretty hard and not every line can be simply rewritten, so creating the same directions is annoying.
20. Day 20 Log (5/29/19): We continued debugging the C++ codes, mainly focusing on how to implement vectors, the C++ version of ArrayList, so we could have an array of undetermined length for our eligibleColumns().
21. Day 21 Log (5/30/19): We continued debugging the C++ code, but we also implemented the Python code into PyCharm so we can take alternate routes.
22. Day 22 Log (5/31/19): Jason continued to fail at correcting a plurality of errors in the arduino IDE, most notably some kind of overflow and something with regards to the brackets.
23. Day 23 Log (6/4/19): We figured out a plurality of errors in the arduino IDE, and there is now an input into the Serial Monitor resemblant of our Python version. There seems to be some input error, but at least the recursion is working.

Final Report

Our original goals for this module were developing a working connect four AI. We also wanted to interface our project with Rob and Liz’s LED board, which would allow us to physically display a user versus computer game.

The biggest problem we encountered early on was programming the recursive aspect of the game in python. The way the AI works is essentially making several moves in the future for both players and analyzing the position to figure out which move leads to the best scored position in the future. Through furious debugging and persistence we realized that variables were being improperly copied when we were making new boards to test future moves and were able to solve the error. Later on we ran into the issue we are still on, which is converting our python code to C++ that can be manipulated on the arduino IDE.

Overall we created a successful connect four AI and familiarize ourselves with many of the elements of C++. We did not quite reach the point where we could interface with Liz and Rob’s physical LED board, but we still built a game that was strong and we could be proud of. We created an AI that could beat us in Connect Four, so in a way the Computer was like a disciple just smacking its mentors in one on one Connect Four.

Throughout the project, our main resource used was the YouTube video linked above. We used it to structure our miniMax function, the function that goes through all the recursion. However, we ran into many issues with the function, but the template helped us create some pseudocode. We used the pseudocode to recreate the function independently, and when we implemented our version of the code, a lot of the errors were resolved. Other than that YouTube video, we used one set of functions we had coded during ProjectEuler to help us create the conditional statements that check if a player has won the game. It checked four numbers in a row horizontally, vertically, diagonally both in a positive and negative slope, and it multiplied them together. We just had to switch it to check the values. Most of the project was coded from scratch.

From this activity we got a refresher on how to run recursion and learned a lot about how AI’s are programmed and optimized for games such as connect four. If we could have done this again perhaps we would have gotten more into C++ more, knowing arduino would come into play later. That or we would have researched sooner how to interface python and arduino. This ended up being a large hurdle that we have not yet overcome, though we have made significant progress for it.

For others who want to pursue similar projects we would urge strongly considering the best way to interface the software of a project with the physical hardware before you even begin coding. In the future we could develop this with a physical board, or someone could develop a vision processing system to view a regular connect four board, interpret the results, and then make moves. Another cool option would be developing a machine learning version and seeing how it did against the AI.