CSE4705 - Artificial Intelligence. Spring, 2017

A Learning Checkers Player

The final project for this course is to develop a checkers player that learns from its experience, then join in a competition with other checkers players developed by your classmates.

While the overall goal in this project is to develop as effective a checkers player as possible (i.e., one that will do well in the final tournament, the specific goal is to have a checkers player that learns from its experience. Because of this, your checkers player will choose its moves based only on a search of the game tree and evaluation of game states. Specifically, encoding of openings, endings, and other move sequences in your player is *not allowed.* There is more about what *is* allowed below.

These projects will be done in teams of three to four students unless you get permission to do otherwise; the expectations will be the same independent of team size.

1 Language and environment

You are free to use whatever language (or languages) you choose in developing your player, but it must be able to compete as a client using the Sam checkers server. (See the document "The Sam Server Protocol" for details on the server.) The "official" rules are available at the English Draughts Association site, `https://sites.google.com/site/englishdraughtsassociation/handbook-2011` with the following exceptions: none of the rules having to do with the physical board will apply, as each player will need to have his or her own board representation and only interact via transmitted moves; a player that makes an illegal move loses immediately without warning; and we will play using a clock–if you go over your allotted time you lose the game (given the clock, ties become unnecessary, as eventually someone will run out).

You may choose to represent the game state however you please, but you will (naturally) need to communicate with the server using its representation.

Wherever you are running your client, it will need to be available at the final competition, which will be held in Laurel Hall. Make sure that is feasible well before the competition. The server is up and running, so you can test that now.

2 Project debrief

Each team will meet with the instructor (scheduled during class time as much as possible) during the final week of the semester. Debrief will be 10 minutes, and will involve discussion of your program design, how your program updates its evaluation function, and how you trained it. (Note: more details to follow.)

3 The tournament

The tournament will be held during final exams. We will have a round-robin tournament (where each client plays each other client), followed by a two-game final for the two top programs (best of three wins, including earlier match). For this to be feasible we will allow three minutes per player in each match.

4 Documentation and deliverables

Each team will submit a report that includes appropriate design docs for your client, a description of the factors that are used in your evaluation function, and a description on how your program was trained and evaluated. It should also include documented code for all of your programs.

This report should describe how your program progressed over time; it should show how its evaluation function changed as a function of the games it played.

5 What is allowed?

Your program will choose moves based on limited lookahead, using a board-evaluation function at the leaves. Your board evaluation should be a function of observable board features that estimates the Minimax value of a state. Samuel used a set of 25 simple board features, plus 12 "binary connective" terms, as his parameters; these provide some good candidates for parameters in your evaluation function.

Starting from an evaluation function with arbitrary coefficients, your client should learn a better evaluation function by playing games. We will look at a number of possible techniques for doing so (Chapter 21 in particular; Samuel also provides information on what he did), but feel free to apply other techniques.

Bottom line: your program can learn to evaluate boards by playing with any other players, but it must always make its own moves when doing so – so Samuel's technique of playing book games would not be allowed, and the use of opening and endgame databases is prohibited.

Endgame variations

Experience has shown that endgames are tough to do with the same evaluation that was used earlier – there are fewer pieces, hence fewer features; different features may have greater value when trying to "finish up", and so forth. We will, therefore, allow programs to change behavior once during a game – from then on, it can use a different board evaluation, and search depth. *As with any board evaluation, it must be based on learned feature weights, not encoded endgame knowledge.* If you choose to use this feature, your client can only change to endgame mode once in a game, and it must remain there until the end.

We will discuss this further in class, but if you have any questions, be sure to ask.

6 Grading

Grades will be assigned based on a linear function of the quality of your report, the debrief, and how well your client does in the tournament, with weights something like .4, .3, and .3 respectively. However, winning the tournament will be given extra weight, so the winner is virtually guaranteed an A.

References

1. A. L. Samuel, Some studies in machine learning using the game of checkers, IBM Journal 3(3), July 1959. pp. 210-229
   (Retrieved from `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5392560`, 4/9/17)

2. Discussion of Samuel's work in Sutton and Barto, Reinforcement Learning: An Introduction
   (Retrieved from `https://webdocs.cs.ualberta.ca/~jonathan/publications/ai_publications/samuel.pdf`, 4/9/17)