Othello Experiments: 0^{th} Iteration

Proof of Concept

V. Markos

September 30, 2022

Abstract

In this brief report we summarize any results obtained from playing Othello using various toy policies against a random agent. Throughout this report, we discuss the proposed experimental setup, possible extensions and present a plan for future iterations.

1 Experimental Setup

In this series of experiments, we have used $N = 3$ different policies, p_i, $i = 1, 2, 3$, each being a refinement of the previous one — all policies are shown in Figure 1. Using each of these, we play $M = 1000$ games against a random agent, with Prudens playing always with black pieces. For each game, we keep track of the time it takes each agent to make their move as well as of the final score of the game. In this context, a game's *score* is defined as the difference between white and black pieces on board. So, positive scores mean that the random agent wins, negative ones that our agent wins and zeros correspond to draws.

2 Results

We present some brief results in Figures 2, 3 and 4. Starting with Figure 2, we present the average scores for all three policies used, with vertical bars denoting standard deviation. We observe there that, as expected, as policies get more refined, the performance of our agent improves significantly. Using p_1, which corresponds to playing at random, naturally leads to an average score close to 0, while p_2, which corresponds to playing at random but for the cases where one can play on the board's corners, leads to a significantly improved performance for the black. Similarly, p_3, which corresponds to playing on corners when possible and, at the same time, avoid playing on cells next to corners also leads to significantly improved performance.

In Figure 3 we present the absolute numbers of wins, draws and loses of all three policies. Verifying results shown in Figure 2, we observe that as the underlying policy becomes more sophisticated, the performance of our agent also improves. In Figure 3, this is reflected mostly on the numbers of wins and loses (blue and red curves, respectively), while the number of draws remains virtually constant.

Finally, Figure 4 demonstrates again the positive correlation between policy sophistication and performance, using score distributions. As demonstrated there, while score distribution for p_1 is relatively symmetric[1], introducing more elaborate heuristics to our agent's policy significantly skews the score distribution towards negative values, i.e., wins for our agent.

3 Discussion and Future Iterations

Given that any policies related to Othello are/will be first-order, it might be a good chance to gather some data about Prudens's performance in larger/more complex and realistic settings. Moreover, future iterations might include:

- More sophisticated policies compared to p_3;
- More "tough" opponents such as a `minimax` agent(utilizing $\alpha - \beta$ prunning, for instance) or (D)NNs;
- Replace the purely symbolic agent used in this set of experiments with a Neural-Symbolic one — as discussed earlier on this week with Marios.

Lastly, we could also consider what other quantities (other than score and win/lose ratio) would make sense to plot and present, so as to sharpen our understanding of any results. Any related resources may be found at: https://github.com/VMarkos/othello-experiments.

[1] Actually, slightly skewed towards positive scores, i.e., loses.

```
@Knowledge
D1 :: legalMove(X,Y) implies
    move(X,Y);
```

(a) Policy p_1.

```
@KnowledgeBase
D1 :: legalMove(X,Y) implies
    move(X,Y);
D2 :: legalMove(X,Y), legalMove(Z,W),
corner(Z,W), -?=(X,Z) implies
    -move(X,Y);
D3 :: legalMove(X,Y), legalMove(Z,W),
corner(Z,W), -?=(Y,W) implies
    -move(X,Y);
C1 :: implies corner(0, 0);
C2 :: implies corner(0,7);
C3 :: implies corner(7,0);
C4 :: implies corner(7,7);
R1 :: legalMove(X,Y), corner(X,Y)
implies move(X,Y);
```

(b) Policy p_2.

```
@KnowledgeBase
D1 :: legalMove(X,Y) implies
    move(X,Y);
D2 :: legalMove(X,Y), legalMove(Z,W),
    corner(Z,W), -?=(X,Z) implies
    -move(X,Y);
D3 :: legalMove(X,Y), legalMove(Z,W),
    corner(Z,W), -?=(Y,W) implies
    -move(X,Y);
C1 :: implies corner(0, 0);
C2 :: implies corner(0,7);
C3 :: implies corner(7,0);
C4 :: implies corner(7,7);
R1 :: legalMove(X,Y), corner(X,Y)
    implies move(X,Y);
R2 :: corner(X,Y), legalMove(Z,W),
    ?isAdj(X,Y,Z,W) implies
    -move(Z,W);


@Procedures
function isAdj(x,y,z,w) {
const X = parseInt(x);
const Y = parseInt(y);
const Z = parseInt(z);
const W = parseInt(w);
return Z-X < 2 && X-Z < 2 && W-Y < 2
    && Y-W < 2 && (X != Z || Y != W);
}
```

(c) Policy p_3.

Figure 1: All three policies used in this series of experiments.

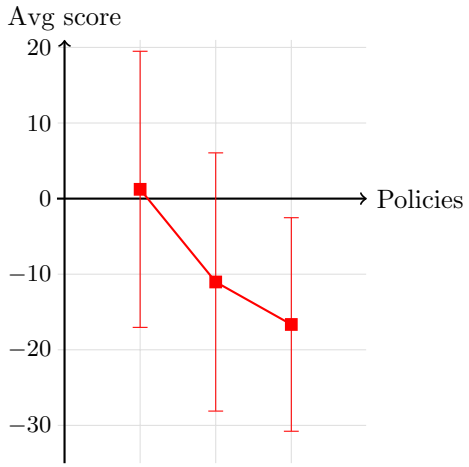

Figure 2: Average score for each policy, p_1, p_2, p_3 (left-to-right). Vertical bars indicate standard deviation.

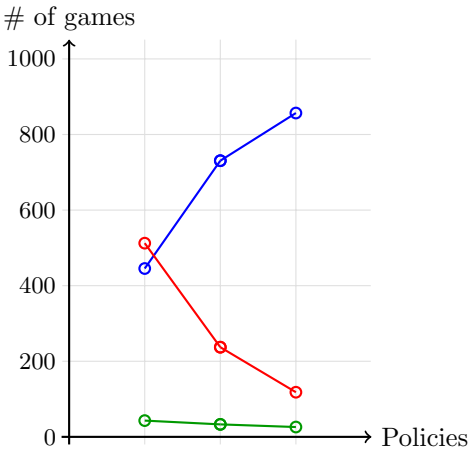

Figure 3: Total number of wins (—), draws (—) and loses (—) with respect to each policy.

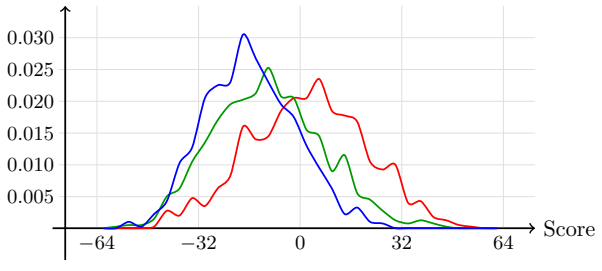

Figure 4: Score distribution per policy. p_1: (—), p_2: (—), p_3: (—).