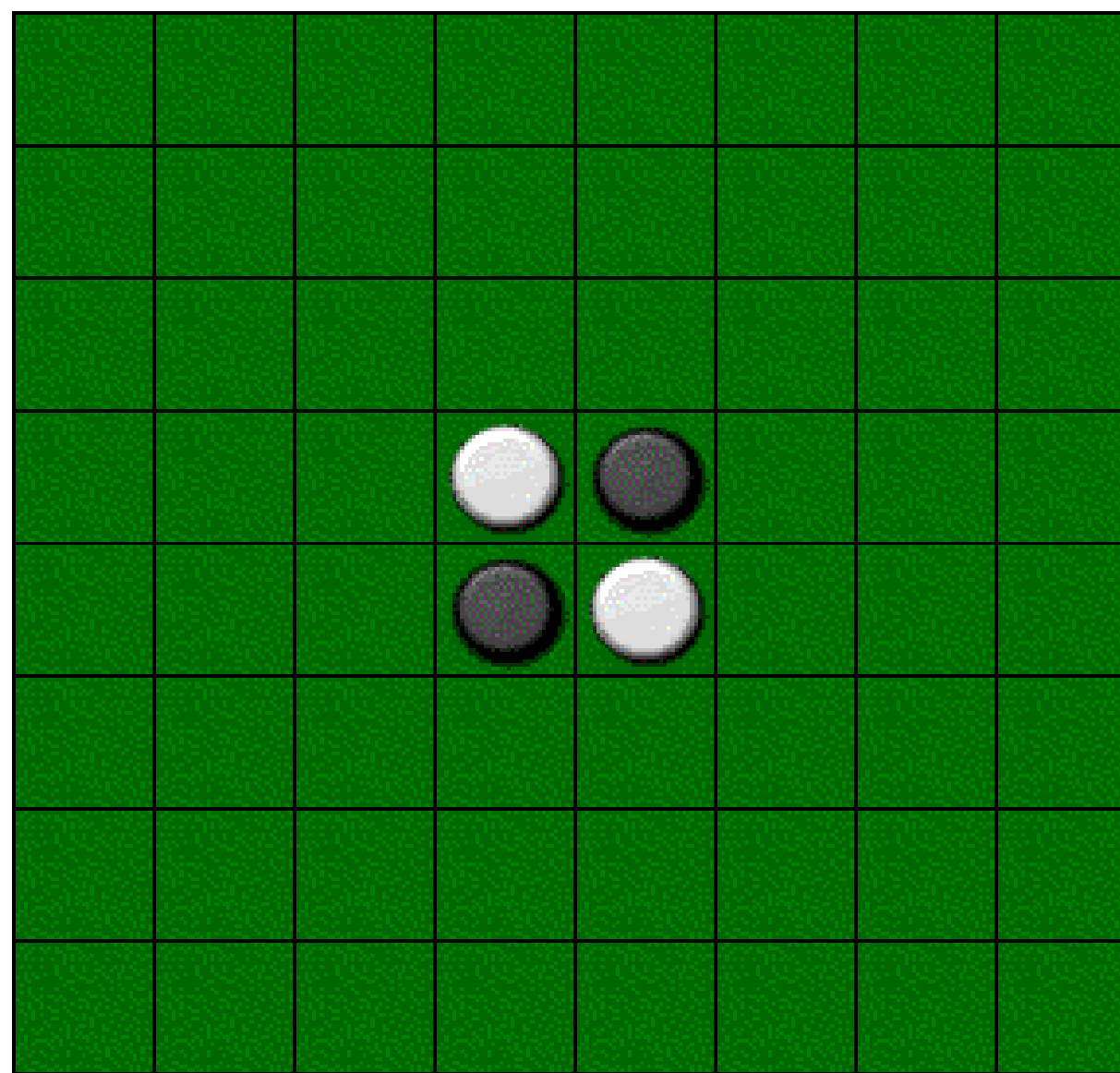


The Use of Genetic Algorithms in Machine Learning; Applications to Othello

Calvin Hayes, June 13, 2006

Background

The project will be to design a program that is capable of playing the board game Othello at a high level. This program will use a genetic algorithm to optimize weights for specific heuristic values that will decide the machine's line of play. Genetic algorithms are becoming more and more popular in the field of artificial intelligence as a way to search many different combinations of settings to find an optimal state. The results of the project can be used to determine how successful this particular algorithm is in setting reasonable heuristic weights that can be used by the computer to play Othello better than a program with these weights set arbitrarily. These results can ideally be generalized to other, real-world uses of this algorithm. To achieve these results, some knowledge of Othello strategy is required, to determine useful heuristics to be weighted. Additionally, other AI techniques, such as minimax trees and alpha-beta pruning will be necessary to help the program run most effectively and efficiently.



Description

Heuristic utility functions were written to detect advantages for the player in several areas. For example, one such function analyzes the number of pieces on the board for the player, as opposed to its opponent; another considers the number of potential legal moves for the player and the opponent on the next turn. The heuristics were combined by making decisions based on a weighted sum of the values returned by each heuristic function. These weights could be set independently and modified at any time. It is the optimization of these weights that is the eventual goal of the genetic algorithm. The genetic algorithm was then put into place, containing 12 'players' (in fact, just combinations of weights) that follow basic rules of Darwinian natural selection, with their winning percentage against a random opponent serving as their evolutionary fitness: propensity for reproduction and a longer lifespan. Eventually the algorithm should result in a 'player' with perfectly balanced weights that are optimal for success in Othello against a random opponent.

```
xterm
-455 -449 913 -348 -444 (moves 46 - 60)
4 4 -6 -20 -14 = -32 | 0 weights 995 451 -781 -121 795 (moves 1 - 15)
-37 -296 -761 -898 -814 (moves 16 - 30)
-763 68 483 -344 240 (moves 31 - 45)
333 -18 765 -831 -237 (moves 46 - 60)
7 14 -14 0 48 = 95 | 5 weights -280 -113 -637 30 42 (moves 1 - 15)
195 -379 76 126 123 (moves 16 - 30)
959 -194 251 524 130 (moves 31 - 45)
-17 689 110 -207 -148 (moves 46 - 60)
mutation - <Function power at 0xb7ae9764> part: 3
mutation - <Function mobility at 0xb7ae379c> part: 0
mutation - <Function mobility at 0xb7ae379c> part: 2
mutation - <Function corner_stability at 0xb7ae9844> part: 3
mutation - <Function edge_stability at 0xb7ae987c> part: 1
mutation - <Function edge_stability at 0xb7ae987c> part: 3
parent 1: 1
parent 2: 3
drop: 11
child: {<Function corner_stability at 0xb7ae9844>: [393, 315, -758, -37], <Function mobility at 0xb7ae379c>: [744, 8, 133, 349], <Function edge_stability at 0xb7ae987c>: [31, 38, 763, 146], <Function power at 0xb7ae9764>: [-665, -277, 471, -368], <Function keysquares at 0xb7ae980c>: [590, 11, 285, -16]}
counter 9
scores
-14 8 57 22 26 = 99 | 18 weights
-334 -439 631 17 -330 (moves 1 - 15)
208 -642 240 41 271 (moves 16 - 30)
931 163 443 566 106 (moves 31 - 45)
-195 585 49 367 429 (moves 46 - 60)
-6 14 14 22 26 = 70 | 14 weights
-687 -342 957 -112 53 (moves 1 - 15)
140 -577 270 -63 220 (moves 16 - 30)
148 9 -942 553 -486 (moves 31 - 45)
-214 238 77 303 -56 (moves 46 - 60)
50 30 -42 6 6 = 50 | 14 weights
-555 -347 875 -4 -79 (moves 1 - 15)
265 -580 240 -239 310 (moves 16 - 30)
352 56 -127 577 -105 (moves 31 - 45)
-199 485 154 354 203 (moves 46 - 60)
6 -28 6 20 18 = 22 | 11 weights
275 951 -431 133 682 (moves 1 - 15)
-510 572 374 368 -251 (moves 16 - 30)
537 239 -595 817 288 (moves 31 - 45)
-254 707 362 40 680 (moves 46 - 60)
-6 4 -18 14 0 = -6 | 9 weights
634 -452 -182 476 -933 (moves 1 - 15)
248 -914 -20 336 111 (moves 16 - 30)
980 -103 600 735 200 (moves 31 - 45)
-143 926 -260 372 548 (moves 46 - 60)
0 -20 22 28 -22 = 8 | 7 weights
251 570 -649 380 470 (moves 1 - 15)
-251 12 165 341 -207 (moves 16 - 30)
587 -289 -308 680 253 (moves 31 - 45)
748 637 201 -81 -383 (moves 46 - 60)
26 26 -12 4 20 = 64 | 8 weights
70 -34 -731 307 398 (moves 1 - 15)
113 -192 274 282 81 (moves 16 - 30)
628 -981 -151 336 162 (moves 31 - 45)
847 632 128 -217 -656 (moves 46 - 60)
14 32 20 4 4 = 74 | 10 weights
-280 -113 -637 30 42 (moves 1 - 15)
195 -379 76 126 123 (moves 16 - 30)
959 -194 251 524 130 (moves 31 - 45)
-17 689 110 -207 -148 (moves 46 - 60)
```

Results

Repeated trials confirmed that the program was, in fact, improving as a result of the genetic algorithm. As expected, drastic improvements over the first 50 iterations gave way to more variable, but steadily increasing results. Important to note, also, are the 'spikes' exhibited when a mutation results in an improved set of weights. Both the average margin of victory of the top 6 and top 3 players showed steady improvement over 400 iterations. A player formed by melding the weights of the top 6 players performed at a level approximately 20 pieces better, on average, than a random opponent. Future experiments could determine whether or not these increases would continue, even if to a lesser amount, over up to 100 iterations.

