Machine Learning – Checkers

Note Title

Arthur Samuel (IBM ≈ 1958)

"Some Studies in Machine Learning Using Checkers"

IDEA: He knew how to play checkers – not an expert

Q: Could he teach a program to learn to play checkers better than he?

Could the program discover better ways to play the game?

Approach: Teach basic Minimax algorithm

Start w/ a basic heuristic

Let machine learn better heuristics (How???)

- play games vs. machine
- have others play game vs. machine

} → record moves + results
⋮
note final outcome of a game

try to develop correlation between moves + outcomes -

= need a good "board evaluation heuristic"

- Obvious: piece advantage: (#your pieces) - (#opponent's pieces)

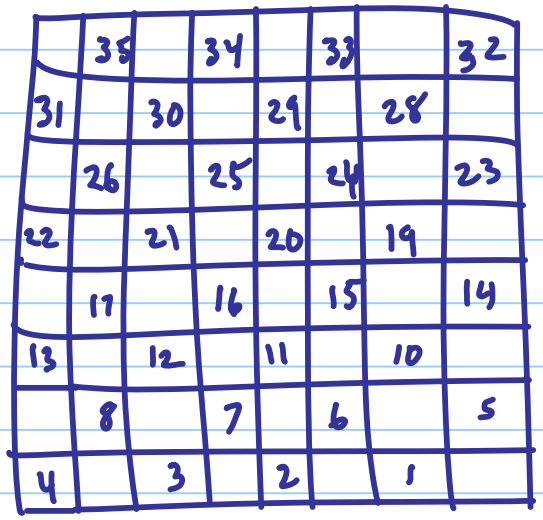
  King advantage: (# your kings) - (# opponent's kings)

Other? - 32 different "scoring functions"

$f_1(\text{Board}) \ldots f_{32}(\text{Board})$

e.g.

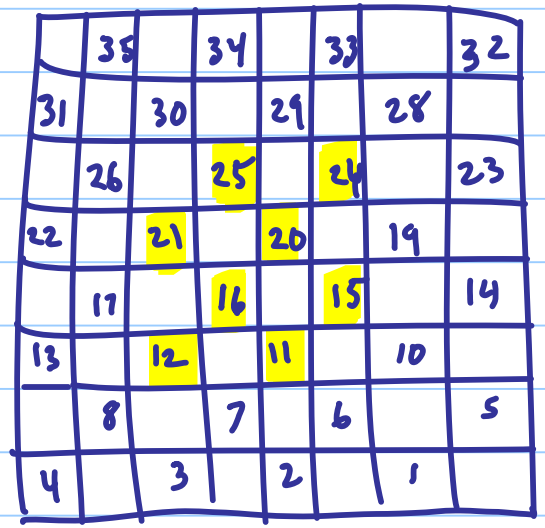
CENT (Center Control I) +1 for each of the following squares 11, 12, 15, 16, 20, 21, 24, 25 occupied by a passive piece



'27' not used

'18' not used

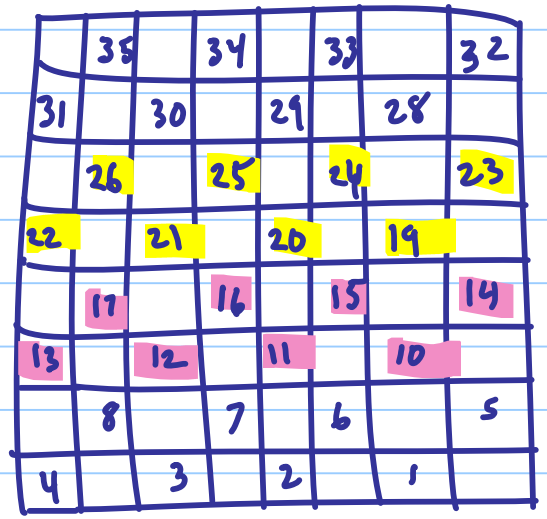
'9' not used



Is this important?
How important?

CNTR (Center Control II) – +1 for each square 11, 12, ... 25 either occupied by active piece or to which active piece can move

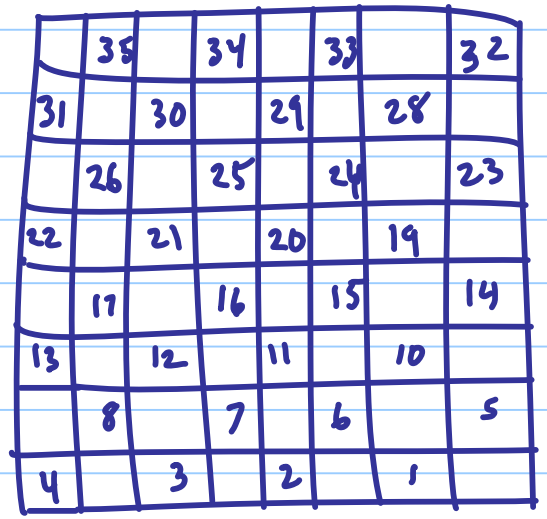
KCENT (King Center Control) occupied by passive king



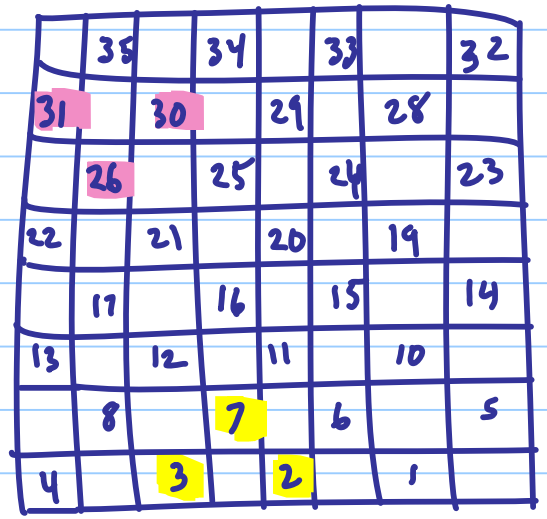
ADV ("advancement")

+1 for each passive piece in 5th + 6th row

-1 for each passive piece in 3rd + 4th row



GAP +1 for each single empty square separating 2 passive pieces on diagonal, or separating passive piece from edge of board



ORED

Triangle of Ored

+1 if no passive kings AND
if Triangle of Ored (2,3,7 for black
26,30,31 for white)

is occupied by passive pieces

"Scoring polynomial" $f(\text{Board}) = \sum_{i \in S} c_i f_i(\text{Board})$ + piece advantage

S is some subset of the 32 functions & c_i are coefficients

How to choose

(a) which functions belong in S ?

(b) what values of c_i correspond to each f_i, $i \in S$?

Can the machine learn a good set of values ??? How?

IDEA: Have program play games against itself —

Assuming each player would use Minimax w/ their own heuristic,

give each side a different heuristic

find out who wins

(a) Let Black play first

(b) Let White play first

if one side wins both games, their heuristic is better.

give this to the "Computer Player" [Champion]

pick another for "Opponent Player" [Challenger]

IDEA: Let Challenger's heuristic be modified from the Champion's heuristic — perhaps by increasing (e.g. doubling) coefficient of lead term

Play again.

[If Challenger wins again, give this heuristic to Champion

Continue until Challenger does not win
[LOCAL OPTIMUM!]

Do something drastic – add/subtract functions from Scoring polynomial

[limit of 10 functions in Scoring Poly]

⋮

Should help find good set of functions

Other IDEAS: For given state, compute f(state)

now, use Minimax to compute backed up value = f'(state)

determine move.

Suppose f'(state) > f(state). Then state is better than we thought – f(state) should be higher

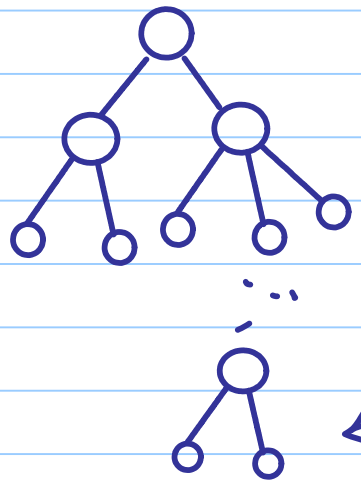
f'(state) < f(state) — state worse, lower value

Other IDEA: When game is over, work backwards to determine whether last move

we made was a good one — certainly not if we lost!
So eliminate this as a potential future choice by modifying scoring polynomials —

If we had 5 possible moves, were any of these better than the one we chose? If so, we should choose it!

Other IDGA (caching)



← evaluate f(state) — has this state been seen before? What was its backed-up value — use it if known.

[Rote learning]

"memorizing facts"

[Learning by generalization]

↓

If we have a table of moves for a bunch of states, can we generalize the move?

"Learning from examples"

↓

given a particular state + "optimal" move,

Can you generalize from one example, two examples, ... ?

effectively looking much further down the tree.

[Learning by experimentation]

↓ (trial + error)

repeat experiment many times

We learn to make a certain move when system is in a certain state

but knowledge is based on experience - not necessarily valid (e.g. based on bad heuristic)

may discover better move

[Sometimes, should make a different move + see what happens

Learning from games played by experts

Play a game by following the moves in a game between two champions.

Score each move w/ your heuristic

But make the move the expert makes.

Count # times you agreed on move (H)

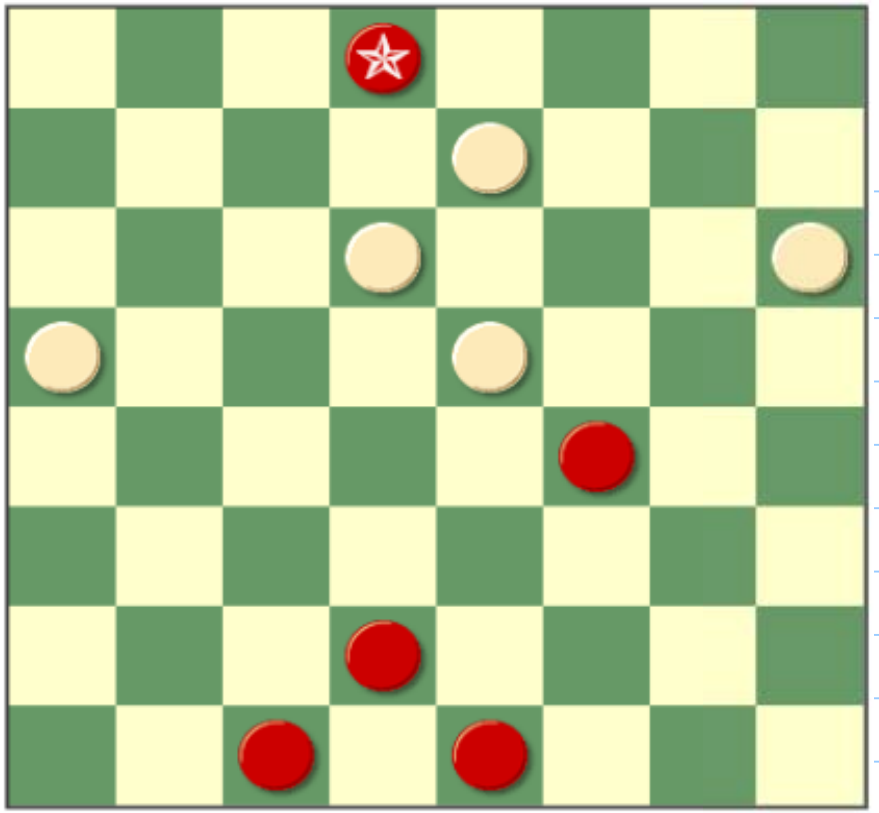
times disagreed (L)

Corellation coefficient: $\rho = \frac{H - L}{H + L}$ $\quad -1 < \rho < +1$

Favor heuristics w/ high value of ρ –

Step through a game played between Samuel's Checkers Program and Robert Nealy:

http://www.fierz.ch/samuel.htm



"Our game...did have its points. Up to [move 16], all of our play had been previously published, except where I evaded "the book" several times in a vain effort to throw the computer's timing off. At the g1-f2 loser and onwards, all the play is original with us, so far as I have been able to find. It is very interesting to me to note that the computer had to make several [very good] moves in order to get the win, and that I had several opportunities to draw otherwise. That is why I kept the game going. The machine, therefore, played a perfect ending without one misstep. In the matter of the endgame, I have not had such competition from any human being since 1954, when I lost my last

See Chinook Project for commentary:
http://webdocs.cs.ualberta.ca/~chinook/project/legacy.html