SOME STUDIES IN MACHINE LEARNING USING THE GAME OF CHECKERS

Arthur L. Samuel (1959)

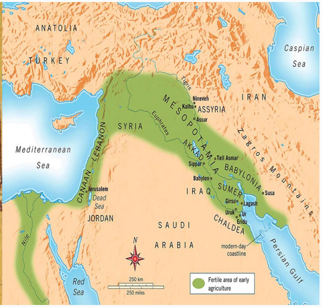
Frank Gabel

April 25, 2019

Artificial Intelligence for Games, Heidelberg Collaboratory for Image Processing

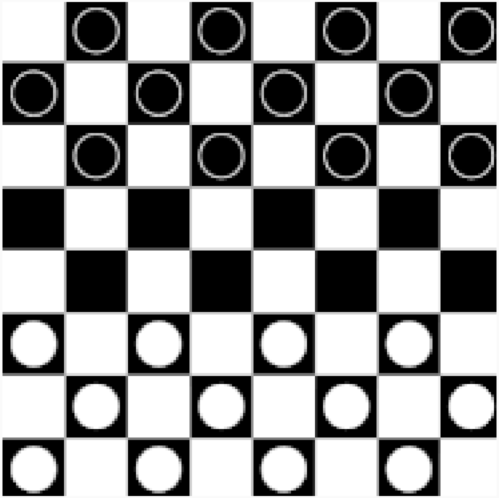
Introduction: A short history of checkers

- Checkers has been played for about 5000 years (even by Plato and Homer)
- It was likely invented in ancient mesopotamia



- Also: the first game in which an AI beat a human ever!

Introduction: The rules of checkers



Checkers from an AI perspective

- Checkers is “simple”: only 10^{20} board positions (Chess: 10^{47}, Go: 10^{250})
- However: computers at the time (1950s) were simple, too: way too complicated to evaluate every possible position
- Still: good candidate for proof-of-concept AIs

Prerequisite: Scoring polynomial

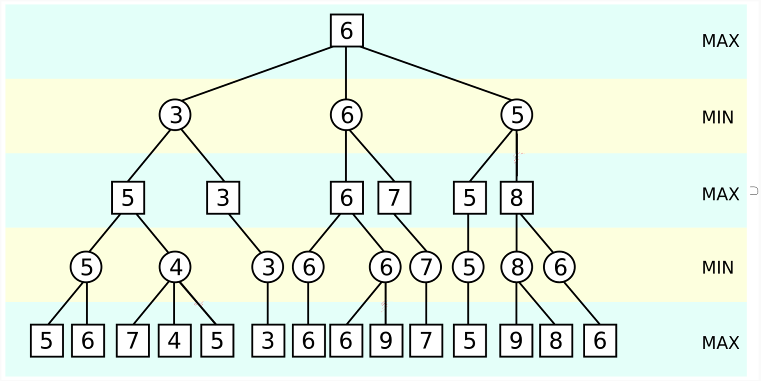
How are the minimax "scores" calculated?

Here: With a "scoring polynomial": $V(x) = a_1 x_1 + \cdots + a_n x_n$
The $x_1, \ldots, x_n$ contain numerical, hand-crafted features, for example:

Ratio	Relative piece/kings advantage
Advancement	The parameter is credited with 1 for each passive man in the 5th and 6th rows (counting in passive's direction) and debited with 1 for each passive man in the 3rd and 4th rows.
Pole	The parameter is credited with I for each passive man that is completely surrounded by empty squares.

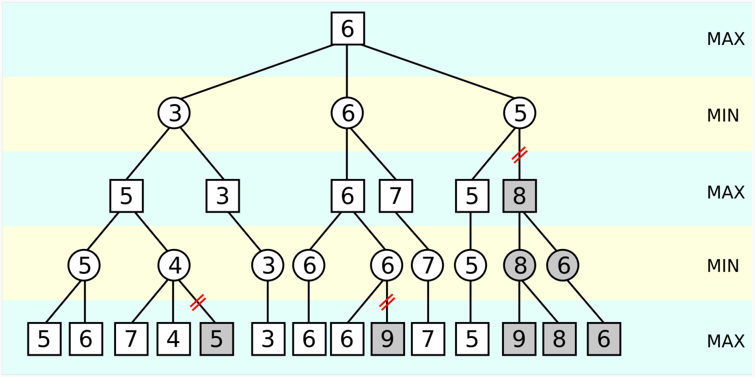
Prerequisite: The MiniMax algorithm

- an algorithm for determining the optimal game strategy for finite two-person zero-sum games with perfect information
- examples: Chess, Checkers, Go, Four Wins, Tic-Tac-Toe



Prerequisite: Alpha–beta pruning

- idea: decrease the number of nodes that are evaluated by the minimax algorithm in its search tree



Samuel’s approaches to learning

Generally:

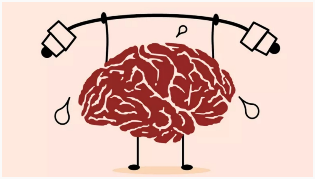
MiniMax + Alpha-Beta Pruning

The “learning approach”:

rote learning + learning-by-generalization

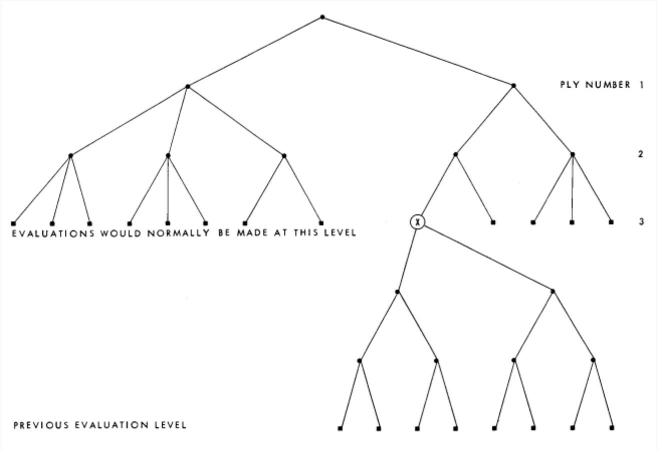
Rote Learning: Idea

- save all of the board positions encountered during play, together with the computed evaluation scores until a depth of e.g. 3
- **learning** by **memorizing**



Rote Learning: Optimizations

- dynamic look-ahead distance
- increase storage capabilities by standardizing board setups


PLY NUMBER 1
2
3
X
EVALUATIONS WOULD NORMALLY BE MADE AT THIS LEVEL
PREVIOUS EVALUATION LEVEL

Rote Learning: Results

After having played “many” games (53,000 board positions saved), Samuel reported:

- a very slow but continuous learning rate most effective in the opening and end-game phases of the play
- the midgame was bad, probably due to a lack of “sense of direction” (i.e. find the fastest way to win)

Generalization: Idea 1

“An obvious way to decrease the amount of storage needed to utilize past experience is to generalize on the basis of experience and to save only the generalizations.”

How do we go about that?

- customize the scoring polynomial of the minimax algorithm (in particular, weights and term removal)
- make the algorithm play against itself and against human opponents
  - **version A** adjusts its coefficients after each move
  - **version B** uses the same evaluation polynomial for the duration of any game
- Then, if **version A** wins, set **version B**=**version A**. Otherwise, randomize **version A** and repeat.

Generalization: Idea 2

“Attempt to make the score, calculated for the current board position, look like that calculated for the terminal board position of the most likely chain of moves.”

How do we go about that?

- Keep track of the difference using a δ
- goal: lower the difference between the calculated goodness of a given board position (according to the evaluation function) and its actual goodness (found through playing out the game to completion)
- optimizations to the technique:
  - replacement scheme
  - binary connective terms

Generalization: Results

First: “tricky but beatable”. . .
Samuel observed the following defects:

- the program was frequently fooled by bad play on the part of its opponent
- too frequent introduction of new terms into the scoring polynomial and the tendency for these new terms to assume dominant positions on the basis of insufficient evidence
- chance wins might result in wrongfully replacement of terms or the whole scoring polynomial

$\Rightarrow$ Optimizing the scoring polynomial is hard!

Discussion and take-aways

- rote learning is an obvious and efficient way to improve the program, but is no real "learning"
- learning-by-generalization was ground-breaking, making the program learn by playing past versions of itself, which would one day be a key component of AlphaGo
- What's next? Samuel, 1969: Some studies in machine learning II

Thanks for your attention. For further details and references, check out my report.