Artificial Intelligence

Week 6: Learning for Game Playing

COMP30024

April 9, 2021

Adversarial Search

- Minimax enumerates all outcomes, selects the best one - inefficient. Two ideas:
- Prune search graph: ignore regions of state space that cannot lead to optimal or complete solutions.
  - Provably ignore suboptimal subgraphs.
- Approximate the expected utility for σ through heuristic evaluation function $f(\sigma)$ to eliminate simulation.
  - Probably ignore suboptimal subgraphs.

Noughts and Crosses

- Branching factor $b = 9$, maximum goal depth is $m = 9$.
- Naively number of possible games $\leq 9! = 362880$.
  - Most games terminate before 9 plys, many identical under board symmetry.
- Complete enumeration of game tree, exact minimax play (TC $\mathcal{O}(b^m)$, SC $\mathcal{O}(bm)$) possible.

Noughts and Crosses

1. Book learning - exact lookup from game tree obtained via minimax. Provably optimal but slowest possible.
2. Efficient α-β ordering:. α-β prevents enumeration of suboptimal regions of state space. Generate optimal successor at each move to maximize efficiency, Approximate optimal ordering via the killer move heuristic:
   - Use iterative deepening to search a small depth below current move, use best recorded path to inform move ordering.
3. Learning weights w_i of evaluation function $f(\sigma) = g\left(\sum_i w_i x_i\right)$. Use f to evaluate the utility for σ at depth limit. Not optimal but lowest time, space complexity.

Games with larger game trees (Chess, Go, etc.) need approximate evaluation.

MENACE

Maintain a set of counters $\{c_1, \ldots c_9\}$ of pseudocounts of state visits. For each game, record all visited states. At the end of the game, for each visited state c_i:

- $c_i \to c_i + \Delta$, if win.
- $c_i \to c_i - \Delta$, if loss.
- $c_i \to c_i$, if tie.

For new games, probability of placing token in square i is

$$p_i = \frac{c_i}{\sum_{j=1}^{9} c_j}$$

MENACE

Number of possible states in noughts and crosses?

- Naïvely $3^9 = 19683$ states (9 squares, 3 possible states for each square).
  - Note number of possible games $>$ number of unique states as a game considers state ordering.
- How to do better? Note game finishes after 9 moves (5 by X, 4 by O).
  - How many ways to place first X on board? $9 = \binom{9}{1}$.
  - How many ways to place first O on board? $72 = \binom{9}{1} \times \binom{8}{1}$.
  - How many ways to place second X on board? $\binom{9}{2} \times \binom{7}{1}$.
  - How many ways to place second O on board? $\binom{9}{2} \times \binom{7}{2}$.
  - How many ways to place third X on board? $\binom{9}{3} \times \binom{6}{2}$.
  - How many ways to place third O on board? $\binom{9}{3} \times \binom{6}{3}$, etc.
  - Sum all up to get 6046
- This can be reduced a bit more via symmetry.

MENACE

1. After a large number of games, the probability of taking a move that leads to a loss should go to zero, and the probability of taking a move that leads to a win should approach 1, so that the agent will favor the optimal action.

MENACE

1. After a large number of games, the probability of taking a move that leads to a loss should go to zero, and the probability of taking a move that leads to a win should approach 1, so that the agent will favor the optimal action.
2. Magnitude of Δ interpreted as 'learning rate' - initially set Δ to move away from uniform probabilities, then anneal the learning rate over training to smaller values to prevent oscillation of probabilities (escaping of local optima) after a large number of games e.g. $\Delta = \frac{C}{1+t}$, for C constant, t number of games played.

MENACE

1. After a large number of games, the probability of taking a move that leads to a loss should go to zero, and the probability of taking a move that leads to a win should approach 1, so that the agent will favor the optimal action.
2. Magnitude of Δ interpreted as 'learning rate' - initially set Δ to move away from uniform probabilities, then anneal the learning rate over training to smaller values to prevent oscillation of probabilities (escaping of local optima) after a large number of games e.g. $\Delta = \frac{C}{1+t}$, for C constant, t number of games played.
3. Probably not a great idea to use MENACE in the project:

MENACE

1. After a large number of games, the probability of taking a move that leads to a loss should go to zero, and the probability of taking a move that leads to a win should approach 1, so that the agent will favor the optimal action.
2. Magnitude of Δ interpreted as 'learning rate' - initially set Δ to move away from uniform probabilities, then anneal the learning rate over training to smaller values to prevent oscillation of probabilities (escaping of local optima) after a large number of games e.g. $\Delta = \frac{C}{1+t}$, for C constant, t number of games played.
3. Probably not a great idea to use MENACE in the project:
   - States are reversible - have to maintain separate counters for each depth of the game tree.

MENACE

1. After a large number of games, the probability of taking a move that leads to a loss should go to zero, and the probability of taking a move that leads to a win should approach 1, so that the agent will favor the optimal action.
2. Magnitude of Δ interpreted as 'learning rate' - initially set Δ to move away from uniform probabilities, then anneal the learning rate over training to smaller values to prevent oscillation of probabilities (escaping of local optima) after a large number of games e.g. $\Delta = \frac{C}{1+t}$, for C constant, t number of games played.
3. Probably not a great idea to use MENACE in the project:
   - States are reversible - have to maintain separate counters for each depth of the game tree.
   - Number of possible states larger - likely no/little data collected on vast majority of positions, leading to uninformed play.