Introduction to AI 2020 spring final project

Author

- 0712238, Yan-Tong Lin
- 0712245, Keui-Chun Kao

The Task

- As in the decription of the assignment
- A variant of orthello that the inner $6 * 6$ board can be placed at any time and the corners of the board are removed

Agent Design

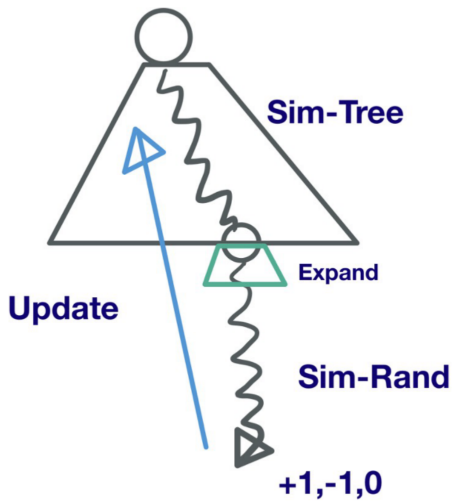
Our agent utilizes both MCTS and Min Max Search to play the designated variant of orthello. We use MCTS as the backbone of our algorithm, and include 3 main improvements to the Vanilla MCTS algorithm, listed as followed.

- Heuristic guided simulation (HS)
- Min Max Agent for closing Game (MC)
- Min Max Search for closing simulation (MS)

the details of MCTS algorithm and the features(HS/MC/HS) will be covered in the following sections.

Basic of Monte Carlo Tree Search

- For each step, MCTS agent do thousands of simulation of the game guided by UCB1 score
- The basic of MCTS can be visualize with the following figure



- we won't cover the very detail of the algorithm please refer to the wiki page

- https://en.wikipedia.org/wiki/Monte_Carlo_tree_search (https://en.wikipedia.org/wiki/Monte_Carlo_tree_search)

- UCB1 score is defined as follow
    - $vis(u) := \#$ of visit to node u
    - $c :=$ a constant of exploration
    - $UCB(u) = \frac{\#\text{ of win}}{vis(u)} + c\sqrt{\frac{\ln(\sum vis(u'\text{s parent}))}{vis(u)}}$
- For each simulation
    - step 0: game tree is clear, only root have been expanded
    - step 1: select (sim-tree)
        - start from root
        - recursively pick the node with highest UCB1 score
        - until reach an unexpanded node
    - step 2: expand (expand)
        - on reaching an unexpanded node
        - expand the node and init its children with default value
    - step 3: rollout (sim-rand)
        - simulate the game to the end by random
    - step 4: backpropagation (upgrade)
        - update the (expanded) nodes on the path with the simulate game result
- after the sumulations, the algorithm return the “most vistied child” of root as the move predicted by the agent

Heuristice Guided Simulation(HS)

- We use a heuristic to guide the roll out process of
- Instead of sampling from uniform distribution, we sample by a heuristic weighted distribution
- The heuristic is defined as follow
    - $H(pos) = 1$, if pos is adjacent to the edges of board
    - $H(pos) = 3$, if pos is on the edges of board
    - $H(pos) = 2$, otherwise

- The implementation detail of HS is as follow
    - $moves$ = all possible moves in the current board
    - $sum = \sum moves[i] * H(moves[i])$
    - $pre[i] =$ prefix sum of $H[i]$
    - sample d randomly with $rand() \bmod sum$
    - find i such that $d \geq pre[i-1]$ and $d < pre[i]$
    - use $move[i]$ as the sampled result

Min Max search Agent for closing Game(MC)

- An improvement is bound to be made if we use exhaustic search instead of probabilistic approximation method(like MCTS)
- But since the seach space is way too large, it's almost impossible to apply exhaustic search at the beginning.
- However, we can mix the two types of agent so that MCTS deals with the earlier stages of games and Min-Max Search deals with the end games.
- This is the MC improvement

A prunning strategy

- A well known prunning strategy called alpha-beta prunning is apllied

Min Max Search for closing simulation(MS)

- Another possible way to improve the MCTS agent is again trying to improve the simulation process
- Using exhaustic search at the end of each simulation should improve the performamce by lowering bias(more closed to the optimal strategy) and variance(since exhaustic search is deterministic)

Experiment

- There are some difficulties to do experiment:
  - since two games requires about $5 * 64 * 2$ seconds to run.
  - and I donnot have a good Win10 system computer at hand.
- So even with a python script, we are not able to conduct too many experiments on the constant c_{UCB1} and experiment on afferent H
- But still a couple of meanningful experiments show positive result on the features(HS/MC/MS) proposed
- In the statement
  - MS agent = Vanilla MCTS + MS improvement
  - MS+HS = Vanilla MCTS + MS improvement + HS improvement
  - MS/HS+HC = game between MS and HS+MC
- Discussions of the experimental result will be cover in detailed in the discussion sections

Time Limit and Agent Strength

- agent strength grows with time limit
- when changing 1.8 second time limit to 4.8 second (vanilla MCTS)
- 4.8 agent had 100% winrate on 1.8 second agent

Heuristice Guided Simulation(HS)

- HS wins 75% of games against Vanilla MCTS
- losing game was when HS took black stone

Min Max search Agent for closing Game(MC)

- HS+MC wins 100% of games against HS counterpart

Min Max Search for closing simulation(MS)

- At the beginning of experiment, MS agent kept exceeding time limit of 5 second since the last iteration can take a lot of time to complete
- by reducing the theshould (i.e. the empty cells count), we finally did successful experiments of MS/HS, MS/HS+HC

- the result shows that MS with time limit lost games to its counterparts

Discussion

Reason of Algorithn designs

Why not deep learning

- I have heard from Dr. I-Chen Wu that Alpha Zero did not perform well on the original game of orthello
- CNN may fail to capture the key information required for the original game of orthello
- I have though of adding hand craft feature(parity…) to improve performance of neural models, but since the lack of time I chose not to risk it.
- There is no data for the variant of orthello, meanning that we can only use alpha zero(means a lot of computational resources is required), or hope fine-tuning can work
- The course is named “Introduction to AI”, study of classical AI algorithms are more to the topic of the course.

Why chose MCTS

- There are two main approaches to reinforcement learning - MC and TD
- MC stands for Monre Carlo
    - Samples from distribution to try to approximate real value
    - High Variance, Low Bias
- TD stands for Temperal difference
    - Use
    - Low Variance, High Bias
- MCTS belongs to the type of MC method
- Since the game of orthello tends to shift a lot between each steps, I do not think that TD method is suitable for the task

The reason for advicing the features

- Heuristic for MCTS roll out
  - Utilizing the domain knowledge that taking edges is usually better.
- Min max for MCTS roll out
  - Using exhaustic search at the end of each simulation should improve the performamce by lowering bias(more closed to the optimal strategy) and variance(since exhaustic search is deterministic)
  - However according to our experiments, when fixing the “thinking time”, the cost the search is not of more value than doing more simulation
- Min max for Closing game
  - Closing game with exhaustic search is better than closing game with MCTS when time limit allows.
  - Can be consider a aggregation of models with discrete flag

A noticible second-go advantage

- Most board games have first-move advantage that can be proven by swapping arguement
- However, orthello, and the variation of it, cannot use the swapping arguement, since redundant move can be shown to be harmful to a player
- It seems that reacting after opponent’s move is a good strategy in orthello.
- This reflects the fact that most match-ups in our experiment have second-go advantage.

Future Work

- More experiments can be conducted to search for optimal c_{UCB1}
- More experiments can be conducted to search for a better H function
- Can still try RL algorithms combined with function approximation strategy(ex: NN) like DQN, A2C, Alpha Zero, etc.

- Learning to Play Othello with Deep Neural Networks
- Application of reinforcement learning to the game of Othello
- Mastering the Game of Go without Human Knowledge