Project Report

Udacity AI nanodegree: Adversarial Game Playing Agent

This project was implemented using MONTE CARLO TREE SEARCH algorithm. The table below shows the performance of Monte Carlo agent against others to compare its performance. Assuming MINIMAX algorithm as a baseline agent for comparisons (since there is no particular requirement for baseline agent).

Agent 1	Competitor	No. of rounds	BA's Success rate (%)	MC's Success rate (%)	Change(%)
Baseline Agent	MINIMAX	50	50	65	15
Baseline Agent	GREEDY	50	65.5	79	13.5
Baseline Agent	RANDOM	50	93.5	91	-2.5
Baseline Agent	SELF	50	50	50.5	Nearly same

All the comparisons are made while the 'fair' flag is active. That means each match will be of two rounds so as to mitigate differences caused by opening position.

(BA - Baseline agent
MC - Monte Carlo
MCTS - MONTE CARLO TREE SEARCH)

Use your results to answer the following questions:

Q1. Choose a baseline search algorithm for comparison (for example, alpha-beta search with iterative deepening, etc.). How much performance difference does your agent show compared to the baseline?

Answer:

- In case of Minimax algorithm there is around **15%** increase in ML agent's performance compared to BA agent's.
- In case of Greedy algorithm there is around **13.5%** increase in ML agent's performance compared to BA agent's.
- Competing with itself the success rate will always around 50%
- And the success rate against RANDOM does not matter because here success/failure is based on luck.

Also one important fact about MCTS is that if we increase the number of rounds success rate will improve

Q2. Why do you think the technique you chose was more (or less) effective than the baseline?

Answer:

- MCTS works very well in games with high branching factor. As it gains information, MCTS increasingly favors moves that are more likely to be good, making its search asymmetric.
- Minimax needs to run to completion to give the best move, which makes its runtime (and run-space) non-flexible. MCTS does not need to run to completion; it outputs stronger plays the longer it runs, but its search can be stopped at any point.
- Minimax function need heuristics to determine which move is good, does not need such a heuristic function.

(*BA - Baseline agent*
MC - Monte Carlo
MCTS - MONTE CARLO TREE SEARCH)