
CSE 537 Artificial Intelligence
Project 2 - Game Search Submission Date - Oct 10, 2015

Student's Details {(Format "<Net_Id>": ("<LastName_FirstName>", "<SBU ID>")}
1) "ADISINGH": ("SINGH_ADITI", "110285096")
2) "AROONWAL": ("ROONWAL_ANSHUL", "110554783")

---

1. Description

Project involves implementation techniques listed below for Adversarial Game- "Connect Four"
1) Minimax search (Question 1)
2) Better Evaluation Function (Question 2)
3) Alphabeta Pruning (Question 3)

Question 1) Minimax search

Running configuration --> run_game(new_player, basic_player)
It was a Tie, but it improved with random player
Execution Time: 49.1597479178
Nodes Expanded: 45661

It improves with--> run_game(basic_player, random_player)
Execution Time: 6.33342726171
Nodes Expanded: 10465

Changes made in these Methods-->
Existing -->

1) minimax(board, depth, eval_fn = basic_evaluate,
get_next_moves_fn = get_all_next_moves,
is_terminal_fn = is_terminal,
verbose = True)

Description--> This method receives a depth of 4 and uses basic_evaluate to generate score.
It calls the minimax_nega_value(board, depth) method

New defined -->

2) minimax_nega_value(board, depth)

Description--> This is a new recursive method. With every recursion, it flips the value/score of the current state so as to alternate levels between Max and Min.
Reference--> Wikipedia for negamax algorithm

3) setnodesExpanded(expandedNodesLocalReceived)
Description--> Setter method for setting the number of nodes expanded.

4) getnodesExpanded()
Description--> Getter method for setting the number of nodes expanded.

5) setnodesExpandedToZeroB4NewGame()
Description--> Setter method for setting the number of nodes expanded to zero before the next search begins.
For instance, first megamax and then AlphaBetaPrunning.

For more detailed description of each of these methods, appropriate comments have been provided in the code.

Question 2) Better Evaluation Function

Calculating the score of each player proportionally ie; I am getting the score as the square of the length of the maximum chain and multilying it with 10 instead of taking just the length. This increases the weigth of the longest chain. I am also calculating the normal distribution of the occupied Cell positions so that we have more moves towards the center are preferred.

Question 3) Alpha Beta Prunning

Running configuration --> run_game(alphabeta_player, basic_player)
Execution Time: 38.8735285196
Nodes Expanded: 48123

It improves with--> run_game(alphabeta_player, random_player)
Execution Time: 5.1599280755
Nodes Expanded: 31024

Changes made in these Methods-->
Existing -->
1) alpha_beta_search(board, depth,eval_fn,
get_next_moves_fn=get_all_next_moves,
is_terminal_fn=is_terminal)
Description--> This method receives a depth of 4 and uses new_evaluate to generate score.
It calls the alpha_beta_negamax(board, depth,eval_fn,alpha, beta) method

New defined -->
2) alpha_beta_negamax(board, depth,eval_fn,alpha, beta)
Description--> This is a new recursive method. It prunes the bad nodes by comparing if alpha > beta. With every recursion, it flips the value/score of the current state so as to alternate levels between Max and Min.
Reference--> Wikipedia for negamax with alpha beta Prunning algorithm

For more detailed description of each of these methods, appropriate comments have been provided in the code.
