CS510 HW Assignment 5

Due: Mar. 13th 11:59pm

Part 1: Programming Assignment

In this assignment you will continue to work on the Othello-playing agent that you created for assignment 4. In the previous assignment, you created a basic agent to play the game. In this assignment, you are going to work to make it play much better. Specifically, you are going to implement two new versions of your Othello agent, the first is a direct improvement of the one you created for the previous assignment, and the second will use a new algorithm.

1.A: Evaluation Function (30 points)

Create a better evaluation function for your current Othello agent. Creating good evaluation functions requires patience and a close analysis of the domain. So, do not hesitate to read about Othello, and about which are the good strategies to win. Play a few games of Othello to familiarize yourself with the game, and then try to think of a good evaluation function. A good evaluation function should return high values for positions where the focus player is likely to win, and low values for positions where it is likely to lose. Be original!

1.B: Monte Carlo Tree Search (70 points)

Create an agent that plays Othello using Monte Carlo Tree Search. Use the following algorithm:

```
MonteCarloTreeSearch(board,iterations):
  root = createNode(board);
  for i = 0...iterations:
    node = treePolicy(root);
    if (node!=null)
      node2 = defaultPolicy(node);
      Node2Score = score(node2);
      backup(node,Node2Score);
  return action(bestChild(root))
```

Where the given functions do the following:

createNode(board): just creates a game tree node from the given board. As explained in class, each node in the game tree stores (at least): its parent, its children, the action that led to this state, the number of times this node has been visited and the average score found so far for this node.

bestChild(node): if the next player to move in node is PLAYER1, it returns the child with the maximum average score, if the next player to move in the node is PLAYER2, then it returns the child with the minimum average score.

treePolicy(node): this function does the following:

- If 'node' still has any children that are not in the tree, then it generates one of those children ('newnode'), it adds 'newnode' as a child to 'node', and returns ‘newnode’.
- If 'node' is a terminal node (no actions can be performed). Then it returns “node”
- If 'node' is not a terminal but all its children are in the tree, then: 90% of the times "nodetmp = bestChild(node)", and 10% of the times "nodetmp = [a child of node at random]" (if you

are curious, this is called an epsilon-greedy strategy). Then, the function returns "treePolicy(nodetmp)"

defautPolicy(node): this function just uses the random agent to select actions at random for each player, until the game is over, and returns the final state of the game.

score(node2): returns the score of the game (you can use the built-in score function in the Othello package, you do NOT need to use the complex evaluation function you created for the previous assignment)

backup(node,score): increments in 1 the number of times "node" has been visited, and updates the average score in "node" with the value "score". If "node" has a parent, then it calls "backup(node.parent,score)".

Compare the performance of this agent against the previous agents you have created. Try running the Monte Carlo Tree Search agent with enough iterations so that it can play at a decent level (values like 10000, 50000, or even 100000 or 1000000 if your implementation is fast enough!).

Part 2: Extra Credit

2.A: Tournament 2 (10 points)
Adapt your two new agents for participating in the Othello tournament, as you might have done for assignment 4. For the Monte Carlo Tree Search agent, instead of receiving the number of iterations, now it receives how much time does it have. Execute as many iterations as you have time to.

2.B: Lisp 2 (10 points)
Reprogram this assignment in Lisp! You'll have to translate the Java Othello code to Lisp, and then your agents.

What to Submit

All homework for this course must be submitted electronically using Bb Vista. Do not e-mail your assignment to a TA or Instructor! If you are having difficulty with your Bb Vista account, you are responsible for resolving these problems with a TA, an Instructor, or someone from IRT, before the assignment it due. It is suggested you complete your work early so that a TA can help you if you have difficulty with this process.

For this assignment, you must submit: Your C/C++/Java/Lisp,/Python/Javascript/... source code, written documentation for your program, and results of your testing. We strongly recommend using a compression utility so that you can compress your files into a single file (with a .zip extension) and just upload it, rather than go through the tedious and error-prone process of uploading each file separately.

Academic Honesty

You must compose all program and written material yourself, including answers to book questions. All material taken from outside sources must be appropriately cited. If you need assistance with this aspect of the assignment, see a consultant during consulting hours.