Project 5: Learning
CS4013/5013: Artificial Intelligence
Due 11:59PM April 5, 2011

Introduction

This project brings you back to mancala where you will use learning to improve your mancala agent, specifically by improving your evaluation function. CS 4013 students will use one learning method and CS 5013 students will use two learning methods.

Because Mancala is actually a solved game[1] for the 4-stone 6-pit version that we used for project 3, we will be modifying the game slightly using the following variations (as suggested in the wikipedia page).

- The number of initial stones in each pit will be a variable, ranging from 4 to 8. You can query GameBoard to find out the number of initial stones. This will be set randomly at the start of each game.
- The ability to capture stones from placing your last seed in an empty pit will be turned off (this will be configurable through the xml files in case you or we need to turn it back on).
- The ability to count the leftover stones at the end of the game as part of your opponent's mancala will also be turned off. Again, this will be controlled through an xml file in case you or we need to turn it back on.

For this project, all students (CS 4013 and CS 5013) will implement decision trees and use them to improve your evaluation functions. There are multiple approaches to using learning to improve your evaluation function and I suggest one here but other approaches are possible! The only rule is that **all** students must use decision trees and CS 5013 students must **also** use another learning method from clustering, regression, or kernel regression.

One approach to using decision trees in your evaluation function would be to have your agent play hundreds of games against other agents (your own and the heuristics) using the rule variations described above. This agent would record features about each game (e.g. by writing out to a text file) that describe the current game situation and the eventual game

[1] http://en.wikipedia.org/wiki/Kalah

outcome. Once sufficient data are acquired, the agent would learn a decision tree to predict the probability of a win at each step and use that tree for the evaluation function. In a similar vein, a CS 5013 student could record data about a specific feature and use clustering to discretize that feature and then feed the cluster data into the tree (in addition to the other features). **Other approaches are possible!**

Extra-credit opportunities

To keep grades within a fair range, all extra credit will be capped at 10 points. This means that no project can receive a grade higher than 110, no matter if they win the ladder, find great exploits, and are very creative.

Wanted dead or alive: bugs or exploits in the simulator

We are reasonably certain that you cannot exploit the simulator. However, any project has bugs and we want to know about them! If you find a bug or an exploit, you can receive extra credit according to the following scale:

- **5 points:** If you find an exploit and report it to us you can receive 5 points extra credit upon verification of the report. Note, we know of two exploits for which you cannot get extra credit as they are already discovered. Since both are both extremely difficult to fix and extremely difficult to implement, I'm not listing them here.
- **10 points:** If you find an exploit and give us a fix for it, you can receive 10 points extra credit upon verification of both the exploit and the fix.
- **1-3 points:** General simulator bugs are much more likely than exploits. Finding a bug and reporting it can get you 1 point. Fixing the bug and giving us the fix (you can't check it in directly but you can give it to us in the bug report) can get you three points. Both bug and fix must be verified for any extra credit to be awarded.

Competition ladder

The class-wide competition ladder will start on March 25th (the day after the project is handed out). You will be ranked by your average score against the other opponents in the ladder.

The first place player will receive one extra point for each night that that player wins the ladder up to a maximum of 5 points. To win, you must be ranked above the random player. The second place player will receive 1/2 extra point for each night that the player is in second place. No player can receive more than 5 extra credit points from the ladder and points will be distributed down the ladder accordingly should the maximum be reached.

Wanted (alive please): creative individuals

Creativity is highly encouraged! To make this real, there are up to 10 points of extra-credit available for creative solutions. Some ideas here include: function approximation (this environment is perfect for function approximation), reinforcement learning, and hierarchical learning (e.g. learning to choose among the high-level behaviors and learning better low-level behaviors).

- Document it in your writeup! I can't very well give extra credit unless I know you did something extra.
- You must still be doing either evolutionary computation
- Remember that by being creative I am referring to the algorithm and not to the ability to creatively download code. All project code must be written exclusively by you except for the sample code that we provide.

Implementation details

You should have your agent collect its data and then learn the tree using a class outside the simulator (e.g. don't have your agent call the tree learning function). Then just have your agent load in the tree. One way to do this is to make use of the Xstream() package that you used on the previous assignment. This gives you each java object serialization but you can also just the default java serialization as well (which saves the objects out to a binary file). Don't go overboard and implement a generalized tree parser! I want you to learn a tree and to use the tree but not to have to go overboard in data structure implementation - make use of what java provides.

Those pesky details

1. Update your aiproject code from project 4. The rule changes described above are not implemented in the earlier versions of the code.

2. We have provided a minimax agent for you to use if you want. This agent only implements minimax and not alpha-beta pruning. Also, because minimax is an answer to a previous assignment, it is available only on D2L (and not in the repository). You can download it from D2L and install it in your package and use it if you don't want to use your minimax client from project 3 (but you are also perfectly welcome to do use your client and it will probably work better since you presumably implemented pruning!).

3. Create a mancala agent that collects learning data, learns a tree, and uses that tree (and other learning if CS 5013) inside the evaluation function, as described above.

4. Ensure that your player runs on linux on the CSN class machines. You can ssh into these machines or go into the lab personally to verify this. Java should be in your path by default. If it is not, java lives in:

   ```
   /opt/java/bin/java
   ```

5. Submit your project on codd.cs.ou.edu using the submit script as described below.

   (a) Log into codd.cs.ou.edu using the account that was created for you for this class. Your username is your 4x4 and your default password is cs#4x4.

   (b) Make sure your working directory contains all the files you want to turn in. Using a similar submit command to the previous projects, you would use the following command to turn in your code. Don't forget to turn in your knowledge file too! Just list it on the command line below as a regular file.

   ```
   /opt/ai4013/bin/submit CS4013 Project5 *.java mancalainit.xml
   ```

   (c) After the project deadline, the above command will not accept submissions. If you want to turn in your project late, use:

   ```
   /opt/ai4013/bin/submit CS4013 Project5Late *.java mancalainit.xml
   ```

6. As with the previous projects, you can test the performance of your agent on the test ladder using the following command. Don't forget to submit your knowledge file too! Just list it on the command line below as a regular file.

   ```
   /opt/ai4013/bin/submit CS4013 Project5TestLadder *.java mancalainit.xml
   ```

Grading rubric

The vast majority of your points go to correctly implementing evolutionary computation. These are broken out as follows. NOTE: CS 5013 students must implement a second approach for one of these categories and compare the results. For example, a CS 5013 student could implement two selection approaches or two fitness functions or two chromosome representations and compare the results in the writeup.

- 50 points for correctly implementing decision trees. A correct decision tree will learn outside of the simulator and will grow a tree using information gain. Your tree can either use probability estimation trees or classification trees (whichever works best for how you put it into the evaluation function).
  - 45 points if there is only one minor mistake. An example of a minor mistake would be a minor error in information gain, off-by-one errors, not correctly calculating probabilities or class labels at the leaves, etc.
  - 40 points if there are several minor mistakes.
  - 35 points if you have one major mistake. An example of a major mistake would be not using information gain correctly (e.g. implementing only the binary version when you have more than binary choices), incorrectly choosing attributes that do not yield the highest gain, etc.
  - 30 if there are several major mistakes.
  - 25 points if you implement some form of learning that turns out a tree but contains enough mistakes that you are unsure exactly how it makes a tree
  - 15 points for an agent that at least does something other than random movements.
- 30/15 (4013/5013) points for correctly taking the trained trees and using them to in your evaluation function
  - 25/10 points for one minor mistake. An example would be correctly bringing the decision tree over (using if/then rules) but then failing to use it to intelligently control your agent (e.g. you never adjust the agent's actions based on the tree or you never adjust the probability thresholds for the agent).
  - 20/8 points for several minor mistakes or one major mistake. An example of a major mistake would be incorrectly bringing the decision tree into the agent.

- For CS 5013 only: 15 points for correctly implementing a second learning method (most likely clustering) and using it somehow in the agent (either by feeding it into the decision tree or by using it directly to guide the agent's actions).
- 10 points: We will randomly choose from one of the following good coding practices to grade for these 10 points. Note that this will be included on every project. Are your files well commented? Are your variable names descriptive (or are they all i, j, and k)? Do you make good use of classes and methods or is the entire project in one big flat file? This will be graded as follows:
  - 10 points for well commented code, descriptive variables names or making good use of classes and methods
  - 5 points if you have partially commented code, semi-descriptive variable names, or partial use of classes and methods
  - 0 points if you have no comments in your code, variables are obscurely named, or all your code is in a single flat method (not sure you can do that with A^* anyway!)
- 10 points for your writeup. A full-credit writeup will describe your implementation of learning, include a learning curve showing how your trees grow, and a discussion of how you used the information from the trees to improve your agent. CS 5013 students must explain their second learning method and how they integrated/used it as well.