Homework 4

15-381/681: Artificial Intelligence (Fall 2017)

Out: November 18, 2017
Due: December 1, 2017 at 11:59PM

Homework Policies

- Homework is due on Autolab by the posted deadline. Assignments submitted past the deadline will incur the use of late days.
- You have 6 late days, but cannot use more than 2 late days per homework. No credit will be given for homework submitted more than 2 days after the due date. After your 6 late days have been used you will receive 20% off for each additional day late.
- You can discuss the exercises with your classmates, but you should write up your own solutions. If you find a solution in any source other than the material provided on the course website or the textbook, you must mention the source. All homeworks (programming and theoretical) are always submitted individually.
- Strict honor code with severe punishment for violators. CMU's academic integrity policy can be found here. You may discuss assignments with other students as you work through them, but writeups must be done alone. No downloading / copying of code or other answers is allowed. If you use a string of at least 5 words from some source, you must cite the source.

Submission

For the written portion, please submit your solutions as a pdf to Gradescope.

For the programming portion, please create a tar archive containing `cfr_plus.py` and `mccfr.py` and submit it to Autolab. The programming portion will be autograded. The command to create the tar file is

```
tar cvzf handin.tar ./mccfr.py ./cfr_plus.py
```

1 Part 1: Written [40 Points]

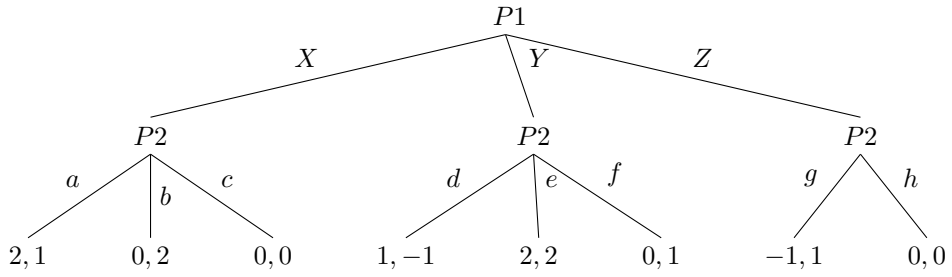
1.1 Nash Equilibrium [10 points]

Calculate the Nash equilibrium of the normal-form game shown below. P_1 chooses the row and P_2 simultaneously chooses the column. The matrix entry $\langle u_1, u_2 \rangle$ shows the value to P_1 and P_2 respectively. *Hint: For each player, write down the expected value equation using variable U for the probability of Up and L for the probability of Left. Take the derivatives and set them to zero. Verify that the resulting probabilities are a Nash equilibrium.*

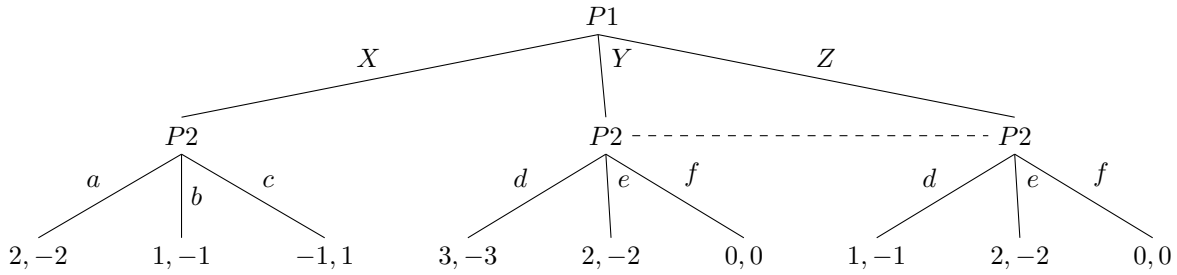
	Left	**Right**
Up	2, 0	-2, 2
Down	1, 1	2, -1

1.2 Extensive Form to Normal Form [10 points]

1. Convert the following extensive-form game to a normal-form game. Solve the extensive-form game via backward induction and give the solution.



2. Convert the following extensive-form game to a normal-form game. Solve the extensive-form game via iterative removal of dominated actions, followed by backward induction, and give the solution.



1.3 Electing the Median [20 points]

The Gibbard-Satterthwaite Theorem proves that in general, any voting system with more than two candidates that is not a dictatorship is susceptible to voters lying about their preferences in order to manipulate the outcome of the election. However, there may be special cases in which a voting rule can be made non-manipulable.

Consider the case of an odd number of students in a classroom that must decide on the temperature of the room. Each student i has a preferred temperature T_i, and wants the winning temperature to be as close as possible to that ideal temperature. That is, if the winning temperature is T^* then student i receives utility $u_i(T^*) = -|T^* - T_i|$.

The students decide to pick the classroom temperature by having everyone vote for a temperature, and then choosing the *median* of those votes. Prove that this voting rule is non-manipulable. That is, prove that student i could not gain by voting for a temperature other than T_i.

2 Part 2: Programming [60 points]

2.1 Description

In this part of the assignment, you will implement Counterfactual Regret Minimization+ (CFR+) and either MCCFR or PureCFR. Your job is to fill out the `solve_game` method in `cfr_plus.py` (which should be CFR+) and `mccfr.py` (which should be PureCFR or MCCFR).

We are providing the file `game.py` for representing the game state. This file contains the `Game` class, which contains a number of helper methods. These will be used for iterating over the game tree. The class also contains a number of print functions that can help you debug and examine the game properties. If you look at the bottom of the `game.py` file, you will see hard-coded equilibrium strategies for the two example games that we are providing. These may be helpful in debugging as well.

The format that you will be using for strategies is described in `cfr_plus.py` and `mccfr.py`.

The two files `coin.txt` and `kuhn.txt` contain a textual representation of two simple extensive-form games. Kuhn is described here: https://en.wikipedia.org/wiki/Kuhn_poker. The value of the game in Kuhn is -0.0555556 and the value of the game in Coin is 0.375. `game.py` contains functions that can print out the tree representation as well as other information.

2.2 Grading

You will implement CFR+ in `cfr_plus.py` and either MCCFR or PureCFR in `mccfr.py`. Autolab will verify that your algorithm converges to a Nash equilibrium (that is, exploitability is close to zero) in a reasonable amount of time. MCCFR/PureCFR should conduct each iteration relatively quickly, since it samples only part of the game tree each iteration. CFR+ should take longer to do each iteration, but should arrive at a more precise solution over time.