CSD 15-784 - Cooperative AI (Fall 2022) Released on: Nov. 3rd, 2022

Homework 2

Maximum: 200 points. **Due on: Nov. 16th, 2022, by 11:59 PM US Eastern time**

Instructions

Show all your work. You may work alone or discuss with **one** other person, but you must follow the following rules or it will be considered cheating. If you discuss with another person, you **must** explicitly acknowledge that specific person on your write-up. Also, the only way in which you may work with another person is to work on a whiteboard together, and then when you are done discussing, to erase the whiteboard, without taking any notes or other record with you, other than what you remember. (Using the zoom whiteboard is allowed if you want to meet remotely.) You should write up your code and your write-up alone.

Please submit the homework on Gradescope!

1 Selfish Procrastination? (50 points)

Researchers A and B are working on a paper together. They have $T = 50$ days left to complete it. They live on opposite sides of the world, so when one is working, one is sleeping. So in fact, there are $2T = 100$ time periods left, with A working in the odd-numbered periods, B working in the even-numbered periods, and the deadline being directly after period 100. There are K units of work left to do on the paper, where $K = 20$. Either researcher can do any unit of work. In any given period, the researcher that is working can choose to do 0 units of work, 1 unit of work, or 2 units of work. Doing one unit of work costs that researcher 1 unit of utility; doing two units of work costs the researcher $2 + \epsilon$ utility, where $\epsilon = 1/100$. However, each researcher will receive $H = 50$ units of utility from all the work for the paper being completed before the deadline (but if the paper is not done by the deadline, all the work is for nothing).

The following is a list of claims about the existence of various kinds of subgame-perfect equilibria. For each, **state** whether the claim is true or false. If it is true, **prove** it by carefully describing the claimed equilibrium. Note that you need to describe the relevant strategies both on and off the equilibrium path of play. (Of course, the game is too large to list all information sets explicitly, but still you can give a complete description of the strategies.) If it is false, **prove** that such subgame-perfect equilibria cannot exist.

Beware: Some of the answers are not immediately intuitive. (At least, on some of the below questions, our own initial inclinations were wrong and careful thinking was required to arrive at the right answer.) We highly recommend writing a dynamic program that helps you find subgame-perfect equilibria of the game. This dynamic program should compute, for increasing values of K and T left, equilibrium strategies and their values to the players. You can potentially compute different equilibria by changing how ties are broken in this dynamic program. (You don't *have* to write any code, though, and in particular you don't have to submit your code.) Please don't rely on code to justify your answers! For example, "There is no such equilibrium, because my code didn't find any." is not a valid answer. It may also help to consider the game for much smaller values of K, T where you can solve it by hand. The below questions are for the above, fixed values of K, T, H, ϵ.

1. There is a subgame-perfect equilibrium in which the paper is not completed.

2. There is a subgame-perfect equilibrium in which (on the equilibrium path of play) neither researcher ever does 2 units of work in one time period.

3. There is a subgame-perfect equilibrium in which (on the equilibrium path of play) there are at most 3 time periods in which exactly 1 unit of work is done.

4. There is a subgame-perfect equilibrium in which Researcher A does more work than Researcher B.

5. There is a subgame-perfect equilibrium in which some 2s come before some 1s, i.e., there is a subgame-perfect equilibrium in which at some time t, the relevant player does 2 units of work and at some later time $t' > t$ the relevant player does 1 unit of work.

2 Sleeping Beauty and Dutch Books (50 points)

Recall the Sleeping Beauty scenario from class. (For the below, Vince's tutorial may also be helpful – in particular, starting at slide 63 / 6:55 in part 4 of the recording on YouTube Vince's explains gives an example of a Dutch book from the literature.)

Imagine that you can offer Sleeping Beauty bets at each point in the scenario, including on Sunday (before the usual scenario starts). Note that we don't want the fact that a bet is offered to reveal information about the day or how the coin landed, so if we offer her a bet when she wakes up, we offer it to her every time she wakes up. Specifically, a set of bets for Sleeping Beauty is specified by the following numbers:

- The payoffs of the bet offered on Sunday: $y_{Su,H} \in \mathbb{R}$ for when the coin comes up Heads and $y_{Su,T} \in \mathbb{R}$ for when the coin comes up Tails.
- Three numbers $y_{Mo,H}, y_{Mo,T}, y_{Tu,T} \in \mathbb{R}$ denoting the payoff of accepting the bet on Monday/Tuesday.

Sleeping Beauty knows all of these payoffs, but of course she doesn't know on Sunday which of $y_{Su,H} \in \mathbb{R}$ and $y_{Su,T} \in \mathbb{R}$ she's going to get if she accepts the bet, and she also doesn't know once she wakes up on Monday/Tuesday which of $y_{Mo,H}, y_{Mo,T}, y_{Tu,T}$ she is going to get if she accepts the bet.

A Dutch book is a set of bets s.t. if you accept all of them, then you lose money with certainty. In the present case, this just means that $y_{Su,H} + y_{Mo,H} < 0$ and $y_{Su,T} + y_{Mo,T} + y_{Tu,T} < 0$, i.e., if you accept both bets whenever they are offered to you, then you will lose money regardless of whether the coin comes up Heads or Tails.

Accepting a Dutch book of bets seems undesirable. After all, when we are offered a Dutch book, we can always walk away with a payoff of 0 by rejecting all bets. Thus, Dutch books are sometimes used as arguments against specific methods of reasoning. If some method X of reasoning sometimes accepts Dutch books, then it seems that we should reject method X. Such arguments are commonly referred to as *Dutch book arguments*. Intuitively, one might think that only the silliest of methods are vulnerable to Dutch books. But in many cases only very specific methods avoid accepting Dutch books. For example, Dutch books alone can be used to justify the use of probabilities and updating beliefs by conditionalization.

In this problem, you will show that if Beauty is not vulnerable to Dutch books, she must have particular types of beliefs. (The assumption throughout is that Beauty is *risk-neutral*, i.e., she attempts to maximize her expected amount of money.)

We will assume that on Sunday, Beauty makes her decision normally, breaking ties in favor of accepting the bet. That is, she accepts the bet on Sunday if and only if $1/2 y_{Su,H} + 1/2 y_{Su,T} \geq 0$.

Thus, for the purpose of this problem, Beauty's beliefs are defined by three numbers $p_{Mo,H}, p_{Mo,T}, p_{Tu,T}$. We require that these sum to 1.

2.1

First we will consider what beliefs Beauty must have if she uses causal decision theory to decide which bets to accept.

Hopefully you remember CDT from lecture. For the purpose of this exercise it is enough to know that CDT accepts the bet if and only if

$$p_{Mo,H}y_{Mo,H} + p_{Mo,T}y_{Mo,T} + p_{Tu,T}y_{Tu,T} \geq 0.$$

(We again assume that ties are broken in favor of accepting the bet.)

Give the set of possible values of $(p_{Mo,H}, p_{Mo,T}, p_{Tu,T})$ s.t. CDT avoids accepting Dutch books. **Prove** that for these values CDT avoids Dutch books and that no other values allow CDT to avoid Dutch books.

Hint: Note that each Dutch book $y_{Su,H}$, $y_{Su,T}$, $y_{Mo,H}$,$y_{Mo,T}$,$y_{Tu,T}$ with $1/2 y_{Su,H} + 1/2 y_{Su,T} \geq 0$ puts a constraint on $p_{Mo,H}, p_{Mo,T}, p_{Tu,T}$. For example, consider $y_{Su,H} = 1, y_{Su,T} = -1, y_{Mo,H} = -2$,$y_{Mo,T} = y_{Tu,T} = 1/2 - \epsilon$. Note that this is indeed a Dutch book and that the Sunday bet is accepted. Thus, for CDT with $p_{Mo,H}, p_{Mo,T}, p_{Tu,T}$ to reject the bet it has to be the case that

$$-2p_{Mo,H} + p_{Mo,T}(1/2 - \epsilon) + p_{Tu,T}(1/2 - \epsilon) < 0.$$

2.2

Second, we will consider what beliefs Beauty must have if she uses evidential decision theory. Again, hopefully you remember this theory and how this works from class, but all you need to know is that EDT accepts the Monday/Tuesday bet if and only if

$$p_{Mo,H}y_{Mo,H} + (p_{Mo,T} + p_{Tu,T})(y_{Mo,T} + y_{Tu,T}) \geq 0.$$

(We again assume that ties are broken in favor accepting the bet.)

Again **give** the set of possible values of $(p_{Mo,H}, p_{Mo,T}, p_{Tu,T})$ s.t. EDT avoids accepting Dutch books. **Prove** that for these values EDT avoids Dutch books and that no other values allow EDT to avoid Dutch books.

3 The Beginnings of a Poker Shark (100 points)

In this problem, you will implement the CFR regret minimizer for sequence-form decision problems.

You will run your CFR implementation on three games: rock-paper-superscissors (a simple variant of rock-paper-scissors, where beating paper with scissors gives a payoff of 2 instead of 1) and two well-known poker variants: Kuhn poker [Kuhn, 1950] and Leduc poker [Southey et al., 2005]. A description of each game is given in the folder cfr_files under the course web page, according to the format described in Section 3.1. That folder also contains a stub Python file to help you set up your implementation.

3.1 Format of the game files

Each game is encoded as a json file with the following structure.

- At the root, we have a dictionary with three keys: `decision_problem_pl1`, `decision_problem_pl2`, and `utility_pl1`. The first two keys contain a description of the tree-form sequential decision problems faced by the two players, while the third is a description of the bilinear utility function for Player 1 as a function of the sequence-form strategies of each player. Since both games are zero-sum, the utility for Player 2 is the opposite of the utility of Player 1.
- The tree of decision points and observation points for each decision problem is stored as a list of nodes. Each node has the following fields

  `id` is a string that represents the identifier of the node. The identifier is unique among the nodes for the same player.

type is a string with value either `decision` (for decision points) or `observation` (for observation points).

actions (only for decision points). This is a set of strings, representing the actions available at the decision node.

signals (only for observation points). This is a set of strings, representing the signals that can be observed at the observation node.

parent_edge identifies the parent edge of the node. If the node is the root of the tree, then it is `null`. Else, it is a pair `(parent_node_id, action_or_signal)`, where the first member is the `id` of the parent node, and `action_or_signal` is the action or signal that connects the node to its parent.

parent_sequence (only for decision points). Identifies the parent sequence p_j of the decision point, defined as the last sequence (that is, decision point-action pair) encountered on the path from the root of the decision process to j.

Remark 1. The list of nodes of the tree-form sequential decision process is given in top-down traversal order. The bottom-up traversal order can be obtained by reading the list of nodes backwards.

- The bilinear utility function for Player 1 is given through the payoff matrix $\boldsymbol{A}$ such that the (expected) utility of Player 1 can be written as

$$u_1(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{y},$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are sequence-form strategies for Players 1 and 2 respectively. We represent $\boldsymbol{A}$ in the file as a list of all non-zero matrix entries, storing for each the row index, column index, and value. Specifically, each entry is an object with the fields

sequence_pl1 is a pair `(decision_pt_id_pl1, action_pl1)` which represents the sequence of Player 1 (row of the entry in the matrix).

sequence_pl2 is a pair `(decision_pt_id_pl2, action_pl2)` which represents the sequence of Player 2 (column of the entry in the matrix).

value is the non-zero float value of the matrix entry.

Example: Rock-paper-superscissors In the case of rock-paper-superscissors the decision problem faced by each of the players has only one decision point with three actions: playing rock, paper, or superscissors. So, each tree-form sequential decision process only has a single node, which is a decision node. The payoff matrix of the game[1] is

$$\begin{array}{c|ccc} & \text{r} & \text{p} & \text{s} \\ \hline \text{r} & 0 & -1 & 1 \\ \text{p} & 1 & 0 & -2 \\ \text{s} & -1 & 2 & 0 \end{array}.$$

So, the game file in this case has content:

```
{
  "decision_problem_pl1": [
    {"id": "d1_pl1", "type": "decision", "actions": ["r", "p", "s"],
     "parent_edge": null, "parent_sequence": null}
```

[1]A Nash equilibrium of the game is reached when all players play rock with probability 1/2, paper with probability 1/4 and superscissors with probability 1/4. Correspondingly, the game value is 0.

```
  ],
  "decision_problem_pl2": [
    {"id": "d1_pl2", "type": "decision", "actions": ["r", "p", "s"],
     "parent_edge": null, "parent_sequence": null}
  ],
  "utility_pl1": [
    {"sequence_pl1": ["d1_pl1", "r"], "sequence_pl2": ["d1_pl2", "p"], "value": -1},
    {"sequence_pl1": ["d1_pl1", "r"], "sequence_pl2": ["d1_pl2", "s"], "value": 1},
    {"sequence_pl1": ["d1_pl1", "p"], "sequence_pl2": ["d1_pl2", "r"], "value": 1},
    {"sequence_pl1": ["d1_pl1", "p"], "sequence_pl2": ["d1_pl2", "s"], "value": -2},
    {"sequence_pl1": ["d1_pl1", "s"], "sequence_pl2": ["d1_pl2", "r"], "value": -1},
    {"sequence_pl1": ["d1_pl1", "s"], "sequence_pl2": ["d1_pl2", "p"], "value": 2}
  ]
}
```

3.2 Learning to best respond

Let Q_1 and Q_2 be the sequence-form strategy polytopes corresponding to the tree-form sequential decision problems faced by Players 1 and 2 respectively. A good smoke test when implementing regret minimization algorithms is to verify that they learn to best respond. In particular, you will verify that your implementation of CFR applied to the decision problem of Player 1 learns a best response against Player 2 when Player 2 plays the *uniform* strategy, that is, the strategy that at each decision points picks any of the available actions with equal probability.

Let $\boldsymbol{u} \in Q_2$ be the sequence-form representation of the strategy for Player 2 that at each decision point selects each of the available actions with equal probability. When Player 2 plays according to that strategy, the utility vector for Player 1 is given by $\boldsymbol{\ell} := \boldsymbol{A}\boldsymbol{u}$, where $\boldsymbol{A}$ is the payoff matrix of the game.

For each of the three games, take your CFR implementation for the decision problem of Player 1, and let it output strategies $\boldsymbol{x}^t \in Q_1$ while giving as feedback at each time t the same utility vector $\boldsymbol{\ell}$. As $T \to \infty$, the average strategy

$$\bar{\boldsymbol{x}}^T := \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}^t \in Q_1 \tag{1}$$

will converge to a best response to the uniform strategy $\boldsymbol{u}$, that is,

$$\lim_{T\to\infty} (\bar{\boldsymbol{x}}^T)^\top \boldsymbol{A}\boldsymbol{u} = \max_{\hat{\boldsymbol{x}}\in Q_1} \hat{\boldsymbol{x}}^\top \boldsymbol{A}\boldsymbol{u}.$$

If the above doesn't happen empirically, something is wrong with your implementation.

Problem 3.1 (25 points). In each of the three games, apply your CFR implementation to the tree-form sequential decision problem of Player 1, using as local regret minimizer at each decision point the regret matching algorithm. At each time t, give as feedback to the algorithm the same utility vector $\boldsymbol{\ell} = \boldsymbol{A}\boldsymbol{u}$, where $\boldsymbol{u} \in Q_2$ is the uniform strategy for Player 2. Run the algorithm for 1000 iterations. After each iteration $T = 1, \ldots, 1000$, compute the value of $v^T := (\bar{\boldsymbol{x}}^T)^\top \boldsymbol{A}\boldsymbol{u}$ where $\bar{\boldsymbol{x}}^T \in Q_1$ is the average strategy output so far by CFR, as defined in (1).

Plot v^T as a function of T. Empirically, what is the limit you observe v^T is converging to?

Your solution should include three plots (one for each game) and three values. Don't forget to turn in your source code too.

★ Hint: represent vectors on $\mathbb{R}^{|\Sigma|}$ (including the sequence-form strategies output by CFR and utility vectors given to CFR) in memory as dictionaries from sequences (tuples `(decision_point_id, action)`) to floats.

★ Hint: in rock-paper-superscissors, v^T should approach the value $1/3$. In Kuhn poker, the value $1/2$. In Leduc poker, the value 2.0875.

3.3 Learning a Nash equilibrium using CFR

Now that you are confident that your implementation of CFR is correct, you will use CFR to converge to Nash equilibrium using the self-play idea described in lecture and recalled next.

The idea behind using regret minimization to converge to Nash equilibrium in a two-player zero-sum game is to use *self play.* We instantiate two regret minimization algorithms, $\mathcal{R}_\mathcal{X}$ and $\mathcal{R}_\mathcal{Y}$, for the domains of the maximization and minimization problem, respectively. At each time t the two regret minimizers output strategies $\boldsymbol{x}^t$ and $\boldsymbol{y}^t$, respectively. Then, they receive as feedback the vectors $\boldsymbol{\ell}^t_\mathcal{X}, \boldsymbol{\ell}^t_\mathcal{Y}$ defined as

$$\boldsymbol{\ell}^t_\mathcal{X} := \boldsymbol{A}\boldsymbol{y}^t, \qquad \boldsymbol{\ell}^t_\mathcal{Y} := -\boldsymbol{A}^\top \boldsymbol{x}^t, \tag{2}$$

where $\boldsymbol{A}$ is Player 1's payoff matrix.

We summarize the process pictorially in Figure 1.

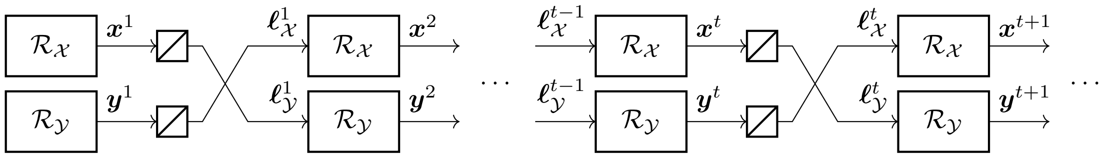

Figure 1: The flow of strategies and utilities in regret minimization for games. The symbol ⊠ denotes computation/construction of the utility vector.

A well known folk theorem establish that the pair of average strategies produced by the regret minimizers up to any time T converges to a Nash equilibrium, where convergence is measured via the *saddle point gap*

$$0 \le \gamma(\boldsymbol{x}, \boldsymbol{y}) := \left(\max_{\hat{\boldsymbol{x}}\in\mathcal{X}}\{\hat{\boldsymbol{x}}^\top \boldsymbol{A}\boldsymbol{y}\} - \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y}\right) + \left(\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y} - \min_{\hat{\boldsymbol{y}}\in\mathcal{Y}}\{\boldsymbol{x}^\top \boldsymbol{A}\hat{\boldsymbol{y}}\}\right) = \max_{\hat{\boldsymbol{x}}\in\mathcal{X}}\{\hat{\boldsymbol{x}}^\top \boldsymbol{A}\boldsymbol{y}\} - \min_{\hat{\boldsymbol{y}}\in\mathcal{Y}}\{\boldsymbol{x}^\top \boldsymbol{A}\hat{\boldsymbol{y}}\}.$$

A point $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$ has zero saddle point gap if and only if it is a Nash equilibrium of the game.

Theorem 1. Consider the self-play setup summarized in Figure 1, where $\mathcal{R}_\mathcal{X}$ and $\mathcal{R}_\mathcal{Y}$ are regret minimizers for the sets $\mathcal{X}$ and $\mathcal{Y}$, respectively. Let $R^T_\mathcal{X}$ and $R^T_\mathcal{Y}$ be the (sublinear) regret cumulated by $\mathcal{R}_\mathcal{X}$ and $\mathcal{R}_\mathcal{Y}$, respectively, up to time T, and let $\bar{\boldsymbol{x}}^T$ and $\bar{\boldsymbol{y}}^T$ denote the average of the strategies produced up to time T, that is,

$$\bar{\boldsymbol{x}}^T := \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}^t, \qquad \bar{\boldsymbol{y}}^T := \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{y}^t. \tag{3}$$

Then, the saddle point gap $\gamma(\bar{\boldsymbol{x}}^T, \bar{\boldsymbol{y}}^T)$ of $(\bar{\boldsymbol{x}}^T, \bar{\boldsymbol{y}}^T)$ satisfies

$$\gamma(\bar{\boldsymbol{x}}^T, \bar{\boldsymbol{y}}^T) \le \frac{R^T_\mathcal{X} + R^T_\mathcal{Y}}{T} \to 0 \qquad \text{as } T \to \infty.$$

Problem 3.2 (25 points)**.** Let the CFR implementation (using regret matching as the local regret minimizer at each decision point) for Player 1's and Player 2's tree-form sequential decision problems play against each other in self play, as described above.

Plot the saddle point gap and the expected utility (for Player 1) of the average strategies $\gamma(\bar{\boldsymbol{x}}^T, \bar{\boldsymbol{y}}^T)$ as a function of the number of iterations $T = 1, \ldots, 1000$.

Your solution should include six plots (two for each game—one for the saddle point gap and one for the utility). Don't forget to turn in your source code too.

★ Hint: represent vectors on $\mathbb{R}^{|\Sigma|}$ (including the sequence-form strategies output by CFR and utility vectors given to CFR) in memory as dictionaries from sequences (tuples `(decision_point_id, action)`) to floats.

★ Hint: to compute the saddle-point gap, feel free to use the function `gap(game, strategy_pl1, strategy_pl2)` provided in the Python stub file.

★ Hint: the saddle point gap should be going to zero. The expected utility of the average strategies in rock-paper-superscissor should approach the value 0. In Kuhn poker it should approach -0.055. In Leduc poker it should approach -0.085.

3.4 Discounted CFR (DCFR)

Problem 3.3 (25 points)**.** Do the task of the previous subsection with DCFR [Brown and Sandholm, 2019] (discussed in class) as the local regret minimizer instead of CFR. Use DCFR parameters $\alpha = 1.5$, $\beta = 0$, $\gamma = 2$.

Your solution should include six plots (two for each game—one for the saddle point gap and one for the utility). Don't forget to turn in your source code too.

3.5 Predictive CFR+ (PCFR+)

Problem 3.4 (25 points)**.** Do the task of the previous subsection with PCFR+ [Farina et al., 2021] (discussed in class) as the local regret minimizer.

Your solution should include six plots (two for each game—one for the saddle point gap and one for the utility). Don't forget to turn in your source code too.

References

H. W. Kuhn. A simplified two-person poker. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games*, volume 1 of *Annals of Mathematics Studies, 24*, pages 97–103. Princeton University Press, Princeton, New Jersey, 1950.

Finnegan Southey, Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes' bluff: opponent modelling in poker. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 550–558, 2005.

Noam Brown and Tuomas Sandholm. Solving imperfect-information games via discounted regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

Gabriele Farina, Christian Kroer, and Tuomas Sandholm. Faster game solving via predictive Blackwell approachability: Connecting regret matching and mirror descent. To appear in the proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2021.