Agents for Playing Normal-Form Games

CS 4320/5314

DUE: Part 1 – 4/30 11:59 PM

Part 2 – 5/10 11:59 PM

**The code and documentation required for this project can be found in the GitHub repository below.**

**https://github.com/osveliz/NFG**

**Goal:** The goal of this assignment is to explore different methods for choosing strategies in normal-form games. You will implement some standard methods, and then implement some strategies of your own.

**Assignment:** The main goal is to develop agents for playing normal-form games. You will be provided with a software platform that provides the infrastructure for generating games and running tournaments between different agents. The platform is capable of generating several different classes of normal-form games including zero-sum games, general-sum games, and “Risk versus Reward” games where some actions give higher expected values, but with larger variation in payoffs, while other actions give a safer but lower payoff.

You will implement several agents which will be tested on all these games, with some complications introduced by uncertainty and repeated play.

Part 1: Simple Agents (DUE 4/30 11:59 PM)

In the first step you will familiarize yourself with the provided code, and implement three specific agents that play according to three concepts that were introduced in class.

**MaxMin Payoff:** This agent should play the strategy that maximizes the minimum possible payoff against any possible opponent action.

**MinMax Regret:** This agent should play the strategy that minimizes the maximum possible regret for any possible opponent action.

**Pure Strategy Nash Equilibrium:**  This agent should play according to a pure-strategy Nash equilibrium (PSNE) of the game if one exists. If there are multiple PSNE, choose the one with the maximum payoff for the player you are choosing a strategy for. If there are no PSNE, play the profile that has the smallest benefit to deviating for any player (i.e., the closest approximate Nash equilibrium.

Part 2: Sophisticated Agents (DUE 5/10 11:59 PM):

In the second part, we introduce two important complications. First, your agent may have uncertainty about the actual payoffs in the game. Second, you may be playing the same opponent repeatedly in the same game, with gives you the opportunity to learn about and respond to their strategy.

**Games with Uncertainty**

The tournament code will generate the game the agents will play as before. However, the game given to each of the agents will be modified within some constraints before it is given to them, so the observed game has some uncertainty. Your agent will be given the parameter settings used to modify the game. This specifies both the maximum number of payoffs that will be modified (chosen at random), and the maximum amount that the payoff could be modified by.

Example:

|  |  |  |
| --- | --- | --- |
|  | L | R |
| U | 1,3 | 4,4 |
| D | 2,3 | 3,1 |

If both players used PSNE the outcome would be (UR) but this game has some uncertainty. The actual game has one outcome with lowered payoffs by a max of 3. In reality the game they are playing is:

|

|

V

|  |  |  |
| --- | --- | --- |
|  | L | R |
| U | 1,3 | 1,2 |
| D | 2,3 | 3,1 |

With the benefit of hindsight, it would have been better for the column player to choose L and therefore the row player to choose D.

**Repeated Games**

Playing repeatedly against the same opponent offers the possibility to learn, both about the true outcomes of the game and about what strategies your opponent is likely to play.

In this setting, after each game your player will be notified the strategy played by your opponent and the expected payoffs of both players. Your agents will then repeat playing the same game against the same opponent nine more times; before swapping positions and playing the same game up to 10 times. The tournament will organize this amongst all agents for all games at numerous parameter settings. Expect these parameters to change.

**Your Agents**

You will design and submit three novel agent designs for playing three different types of games:

1) One-shot games with no uncertainty (as in part 1)

2) One-shot games with uncertainty

3) Repeated games with uncertainty

You must submit and document at least one agent for each of these categories, and explain what your approach is and why it is interesting. For case #2 you must implement some approach for dealing with the uncertainty in the game, and for case #3 you must include some element of learning/adapting to the previous game outcomes in your agent design. We encourage you to develop several possible strategies, test them out in your own tournaments, and select the best one to submit.

You will write a brief report (3-5 pages) that clearly documents the ideas behind the strategies for 1-3, as well as the results of your own testing tournament for each of the three cases. These should play your agents against at least the 3 baseline agents you developed in part 1, and possibly some other alternative agents you have designed. You will be evaluated both on the creativity and thought put into the agents, as well as how well they perform in the overall class tournament.

**Tips:**

* Have your agent make decisions depending upon depending on the amount of uncertainty parameter in the game. For example, if there is no uncertainty in the parameters versus if there is a high level of uncertainty versus a small level of uncertainty your agent should behave differently.
* Your agent may be playing in different types of games as described initially; your strategy might want to take this into account.
* Use what you know about probability to update your beliefs of the game and your opponent.
* Learn from your opponent. If you’re playing against SolidRock then using PSNE is likely not a good best response.
* Consider exploring the game if you have some uncertainty during the first few rounds (the game master will provide you the correct expected payoff which you can use to update your game) and then try to exploit the best outcomes.
* If one strategy does not appear to be performing well against an opponent, try switching to another one.
* Simulate what a few rounds of back-and-forth play might look like before making a decision.

**Deliverables:**

* A short report documenting the design of your novel agents, and the results of a sample tournament as described above.
* Java files implementing the three required agent strategies (initial submission, along with three novel agents of your own design (final submission).

*Put all of your agent files and the report into a single zip archive and upload it to blackboard.*

You will be graded on correctly implementing the three required agents, implementing three non-trivial intelligent agents of your own design, and running the sample tournament and documenting your results. In addition, the top five agents in two different categories in the tournament will earn bonus points:

1) The agents with the highest average payoff

2) The agent with the highest **minimum** score that finish in the **top half** of agents in average score

If you have questions about the assignment or the provide code, you can contact either Dr. Kiekintveld or Oscar Veliz ([osveliz@utep.edu](mailto:osveliz@utep.edu)).