6CCS3AIN & 7CCSMAIN, 2016, Coursework 3 (Version 1)

1 Introduction

This coursework exercise asks you to implement versions of Axelrod's Iterated Prisoner's Dilemma tournament. Part of the coursework is learning about this tournament (we did not cover it in lectures).

This exercise will be assessed.

2 Getting started

You should download the file `axelrod-cooperation.pdf` from from the KEATS page (near where you found this document). This is a copy of the paper:

> Robert Axelrod and William D. Hamilton, The Evolution of Cooperation, *Science*, **211**(4489):1390–1396, 1981.

You need to read this short paper.

3 What you need to do

This coursework requires you to do five things:

1. Write a Netlogo model in which agents interact to play the Prisoner's Dilemma game. This should include at least two approaches that agents can adopt to play the iterated Prisoner's Dilemma.
2. Evaluate the performance of the approaches to playing iterated Prisoner's Dilemma over several tournaments.
3. Extend your Netlogo model to create an evolutionary tournament in which agents select their approach to playing the iterated Prisoner's Dilemma based on the past performance of the approaches.
4. Evaluate the the performance of the approaches over several evolutionary tournaments.
5. Introduce at least one more game in addition to the iterated Prisoner's Dilemma, and run an evolutionary tournament of this game.

Check Section 3.6 for a list of the limitations on what you are allowed to do with the code.

I suggest that you read through the whole of this document before starting on the coursework.

3.1 Iterated Prisoner's Dilemma tournament

Your model needs to be one in which you have a number of agents, and in which those agents interact with each other over a number of rounds. You are pretty much unrestricted in how you make this happen, but the following thoughts might be useful.

- I think there are two obvious ways for the interactions to happen:
    1. Have the agents be turtles that move aorund randomly and interact with other agents on the same patch, or with other agents within some distance of each other.

2. Have the agents be patches that interact with their neighbours (either four or eight neighbors).

   Whichever of these you use, you probably want to have the Netlogo world wrap both horizontally and vertically.

- To implement Tit-for-Tat, and other more sophisticated strategies, agents will need to recod their previous interactions with other agents and remember which interactions are with which agents.
- The payoffs for the Prisoner's Dilemma in the Axelrod game are a bit different from the ones I used in the lecture. You can use either.

The things that you must include are:

1. At least two approaches to playing the game, one of which must be Tit-for-Tat from the Axelrod paper.

   You are allowed to implement as many approaches as you want, and the other(s) can be one(s) you make up, or one(s) from the Axelrod paper.
2. A way for the user to select how many agents use each approach to playing the game.
3. Some visual method to indicating which agents are using which approach (for example by changing the colour of the agent).
4. Something which indicates the score of all the agents using a particular approach.

I am not going to dictate how you evaluate the tournament, or how you establish the winner. You decide and I'll judge how good a job you did when I read your report.

3.2 Evolutionary tournament

In the section with the title "Robustness", Axelrod and Hamilton describe, very briefly, another kind of tournament which they ran, they say:

> The ecological approach takes as given the (approaches to playing the game) which are present and investigates how (these approaches) do over time when interacting with each other. This analysis was based on what would happen if each of the (approaches) were submitted to (a new tournament) in proportion to its success in the previous (tournament).

Parentheses mark where I have replaced words from Axelrod and Hamilton with terms that I have been using here.

So the ecological approach takes some set of agents, each taking some approach to playing the game (with multiple agents typically taking the same approach) and runs a number of interactions (a tournament). Then, based on how well each approach has done (for example calculating the average score of each agent playing that approach), a new set of agents is generated such that the number playing approach A is proportional to the average score of A in the previous tournament. This process is repeated some number of times.

This is what you have to implement, and then evaluate by running some evolutionary tournaments. The things that you must include are:

1. Some interface widget that allows the user to decide whether to run an evolutionary tournament, or whether to run just as single tournament.
2. Some method to indicate the number of agents using each approach at each round of the evolutionary tournament.

Again, I am not going to dictate how you evaluate the evolutionary tournament, or how you establish the winner. You decide and I'll judge how good a job you did when I read your report.

3.3 A second tournament

You now have to run a tournament that uses an iterated version of a game other than the Prisoner's Dilemma. You can use any game you want, including ones you make up. Some examples are in the lecture slides. Other examples, which are both similar in payoff structure to the Prisoner's Dilemma, but are different enough to have been studied in some detail in their own right, are:

- The Stag Hunt, which has a payoff matrix like:

		i	
		defect	coop
j	defect	2 2	1 3
	coop	3 1	4 4

  The difference from the prisoner's Dilemma is that now it is better if you both co-operate than if you defect while the other co-operates.

- Chicken, which is captured by the payoff matrix:

		i	
		defect	coop
j	defect	1 1	2 4
	coop	4 2	3 3

  Here the difference with the Prisoner's Dilemma is that mutual defection is the worst case outcome.

3.4 Other things to do

You could, in addition, use more than two approaches to playing the games, and you could write code for, and run tournaments with, additional games. You could also try more than one method for deciding the proportion of agents using each approach in the evolutionary tournament. What I described above is a global approach, a more local approach would have each agent look at the scores of the agents around it and use that to choose which approach to use.

Any such work will get credit.

3.5 Write a report

Write up a description of your program along with your evaluation of the tournaments in a separate report that you will submit along with your code.

As you work through an implementation of the tournaments, you will find that you are making lots of decisions about how precisely to translate your ideas into working code. The report should explain these at length. The perfect report will give enough detail that I don't feel I have to read your code in order to understand what you code does (I will read it anyway).

Remember, when doing this, that there is credit for creative and beautiful solutions. Make sure you highlight these aspects of your work, especially things that make your work unique.

Having said that, reports that are needlessly long will not get any more credit. I value concise reports (I have to read around 90 of them).

Your report should also analyse the outcomes of the tournaments. Your analysis should be based on the evaluation, and all your conclusions should be justified by the data that you have collected.

3.6 Limitations

In comparison with previous courseworks, there are not many limitations.

There is one, however, that I will state despite the fact that it should be obvious: you are not allowed to copy code that you might get from other students or find lying around on the Internet.

The iterated Prisoner's Dilemma has been widely studied, and there are lots of implementations out there. There are certanly Netlogo implementations. Copying code from any of them is plagiarism, and we will be checking your work for it.

Other limitations are:

1. You are still restricted to just submitting a single `.nlogo` file. This means that using extensions is not recommended.

   As I understand it, extensions are implemented in a distribution-dependent way. That means that if you use them, they may not work on my machine, and if they don't, you will lose credit because your code does not run.

4 What you have to hand in

You should hand in your NetLogo program and your report.

The file containing your program should be named:

`ipd-<lastname>-<firstname>.nlogo`

so my program would be named `ipd-parsons-simon.nlogo`.

Your report must be a PDF document, and should be named:

`ipd-<lastname>-<firstname>.pdf`

The names you use must be the names under which you are registered for the module.

These two files should be submitted individually through KEATS (do not combine the files into a ZIP archive).

5 How your work will be marked

There will be six components of the mark for your work:

1. Functionality

   As discussed above, your code is required to provide:

   (a) Two or more approaches to paying the iterated Prisoner's Dilemma, including Tit-for-Tat.

   (b) The ability to run iterated Prisoner's dilemma tournaments.

   (c) The ability to run an evolutionary tournament.

(d) An implementation of at least one game in addition to Prisoner's Dilemma that can be used for evolutionary and non-evolutionary iterated game tournaments.

2. Style

   There are no particular requirements on the way that your code is written this time, but it should follow standard good practice in software development.

3. Documentation

   All good code is well documented, and your work will be partly assessed by the documentation you provide.

   Netlogo offers two approaches to documentation, through the use of comments in the code, and through the use of the Info tab. You should use both.

4. Report

   The contents of the report are described above. Your work will be assessed against the criteria laid out there.

5. Results

   In addition to looking at your experimental results to assess the functionality of your code, I will be looking to check that you did some evaluatione, that you analysed the data as required, and that you have drawn sensible conclusions from the experiments. Proper statistical analysis will be credited.

6. Creativity

   I was pretty amazed by some of the creative approaches that people came up with for the first coursework. However, there was no scope in the marking scheme to recognise that creativity. For this coursework, there will be a mark component to recognise creative and beautiful solutions.

   As with all instances of beauty, I can't specify what this is, other than to say that I will know it when I see it. Impress me.

6 Remember

Netlogo is a free download. If you want to obtain a copy to run on your own computer, please go to:

https://ccl.northwestern.edu/netlogo/download.shtml

and follow the instructions to download version 5.3.1.

Version 5.3.1 is the one that is running in the labs, and is the version that will be used to run your code for the assessed exercises.

7 Version list

- Version 1, November 20th 2016