Axelrod1980_solution

January 22, 2016

0.1 Solution of Axelrod 1980

In [1]:
```
import numpy as np
```

0.1.1 Implement the five strategies

In [2]:
```
# We are going to implement five strategies.
# Each strategy takes as input the history of the turns played so far
# and returns 1 for cooperation and 0 for defection.

# 1) Always defect
def always_defect(previous_steps):
    return 0

# 2) Always cooperate
def always_cooperate(previous_steps):
    return 1

# 3) Purely random, with probability of defecting 0.5
def random(previous_steps):
    if np.random.random(1) > 0.5:
        return 1
    return 0

# 4) Tit for tat
def tit_for_tat(previous_steps):
    if len(previous_steps) == 0:
        return 1
    return previous_steps[-1]

# 5) Tit for two tat
def tit_for_two_tat(previous_steps):
    if len(previous_steps) < 2:
        return 1
    # if the other player defected twice
    if sum(previous_steps[-2:]) == 0:
        # retaliate
        return 0
    return 1
```

0.1.2 Write a function that accepts the name of two strategies and competes them in a game of iterated prisoner's dilemma for a given number of turns.

In [3]:
```
def play_strategies(strategy_1, strategy_2, nsteps = 200):
    # The following two lines are a bit complicated:
    # we want to match a string (strategy_1) with a name of the function
    # and the call globals()[strategy_1] does just that. Now
    # pl1 is an "alias" for the same function.
    pl1 = globals()[strategy_1]
    pl2 = globals()[strategy_2]
    # If you prefer, you can deal with this problem by using
    # a series of if elif.

    # Now two vectors to store the moves of the players
    steps_pl1 = []
    steps_pl2 = []
    # And two variables for keeping the scores.
    # (because we said these are numbers of years in prison, we
    # use negative payoffs, with less negative being better)
    points_pl1 = 0
    points_pl2 = 0
    # Iterate over the number of steps
    for i in range(nsteps):
        # decide strategy:
        # player 1 chooses using the history of the moves by player 2
        last_pl1 = pl1(steps_pl2)
        # and vice versa
        last_pl2 = pl2(steps_pl1)
        # calculate payoff
        if last_pl1 == 1 and last_pl2 == 1:
            # both cooperate -> -1 point each
            points_pl1 = points_pl1 - 1
            points_pl2 = points_pl2 - 1
        elif last_pl1 == 0 and last_pl2 == 1:
            # pl2 lose
            points_pl1 = points_pl1 - 0
            points_pl2 = points_pl2 - 3
        elif last_pl1 == 1 and last_pl2 == 0:
            # pl1 lose
            points_pl1 = points_pl1 - 3
            points_pl2 = points_pl2 - 0
        else:
            # both defect
            points_pl1 = points_pl1 - 2
            points_pl2 = points_pl2 - 2
        # add the moves to the history
        steps_pl1.append(last_pl1)
        steps_pl2.append(last_pl2)
    # return the final scores
    return((points_pl1, points_pl2))
```

In [4]:
```
play_strategies("random", "always_defect")
```

Out[4]: (-516, -168)

0.1.3 Implement a round-robin tournament, in which each strategy is played against every other (including against itself) for 10 rounds of 1000 turns each.

```
In [5]: def round_robin(strategies, nround, nstep):
            nstrategies = len(strategies)
            # initialize list for results
            strategies_points = [0] * nstrategies
            # for each pair
            for i in range(nstrategies):
                for j in range(i, nstrategies):
                    print("Playing", strategies[i], "vs.", strategies[j])
                    for k in range(nround):
                        res = play_strategies(strategies[i],
                                              strategies[j],
                                              nstep)
                        #print(res)
                        strategies_points[i] = strategies_points[i] + res[0]
                        strategies_points[j] = strategies_points[j] + res[1]
            print("\nThe final results are:")
            for i in range(nstrategies):
                print(strategies[i] + ":", strategies_points[i])
            print("\nand the winner is....")
            print(strategies[strategies_points.index(max(strategies_points))])
```

```
In [6]: my_strategies = ["always_defect",
                          "always_cooperate",
                          "random",
                          "tit_for_tat",
                          "tit_for_two_tat"]
```

```
In [7]: round_robin(my_strategies, 10, 1000)
```

```
Playing always_defect vs. always_defect
Playing always_defect vs. always_cooperate
Playing always_defect vs. random
Playing always_defect vs. tit_for_tat
Playing always_defect vs. tit_for_two_tat
Playing always_cooperate vs. always_cooperate
Playing always_cooperate vs. random
Playing always_cooperate vs. tit_for_tat
Playing always_cooperate vs. tit_for_two_tat
Playing random vs. random
Playing random vs. tit_for_tat
Playing random vs. tit_for_two_tat
Playing tit_for_tat vs. tit_for_tat
Playing tit_for_tat vs. tit_for_two_tat
Playing tit_for_two_tat vs. tit_for_two_tat

The final results are:
always_defect: -90082
always_cooperate: -90132
random: -84673
tit_for_tat: -75001
tit_for_two_tat: -77549
```

```
and the winner is...
tit_for_tat
```

In [ ]: