Università degli Studi di Bari
Dipartimento di Informatica

LACAM
Machine Learning

Iterated Pysoner’s Dilemma

Or how an altruistic behaviour can emerge by iterating selfish games

antonio vergari - *an irreducible reductionist*

July 27, 2015

Context & Background

Prisoner’s Dilemma

The ***Prisoner's Dilemma*** (**PD**) is a two-players perfect information game where each player is asked to do a single decision: **C**ooperate or **D**efect, according to a payoff matrix:

	C	**D**
C	(R, R)	(S, T)
D	(T, S)	P, P

for any payoff combination $T > R > P > S$.

The Nash equilibrium is P, P, the Pareto's one is R, R.
There is no need to simulate anything to know a *rational agent strategy*.

Iterated Prisoner’s Dilemma

For $2R > S + T$, the ***Iterated Prisoner's Dilemma*** (**IPD**) is a sequence of N PD Games. Each player's outcome is the sum of the N payoffs received.

Again, a rational player would always **D**efect (it can be proved inductively this is the dominant strategy).

However, some key factors make it differ from a simple PD:

- more than four strategies
- possibility to change strategies on new information
- evironment (other players strategies) adds variability

Why and what to simulate

While, given the player strategies, one can predict the outcome of a single IPD game between two players, it is not easy to predict the global outcome of *several IPD games amongst all players* (it can be computed as the sum of payoffs received along all games).

The outcome now depends on the **payoff matrix**, **player histories**, and **player population** characterization.

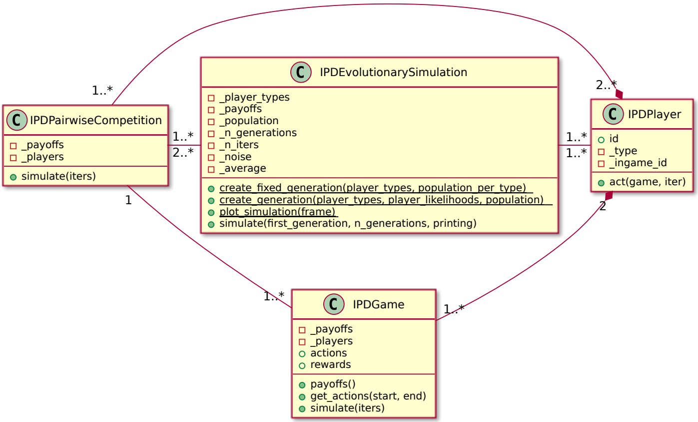
Other notions of *bounded rationality* can complicate things even more.
Moreover, remind the notion of Hofstader's *superrationality*

Following Axelrod's road two kinds of simulations are presented:

- simulating exhaustive pairwise IDP Games for a population of players
- simulating the evolution of player strategies a fixed size population across generations

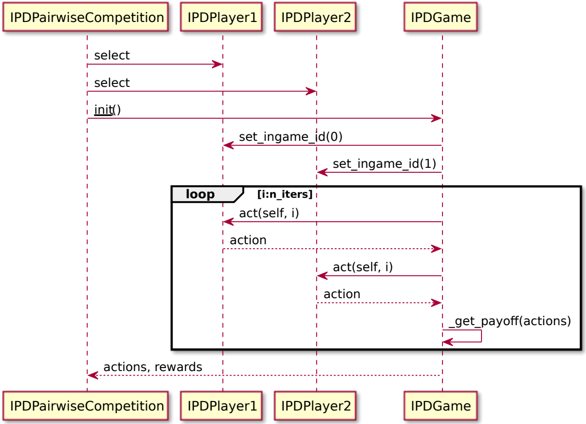
Modeling

Components

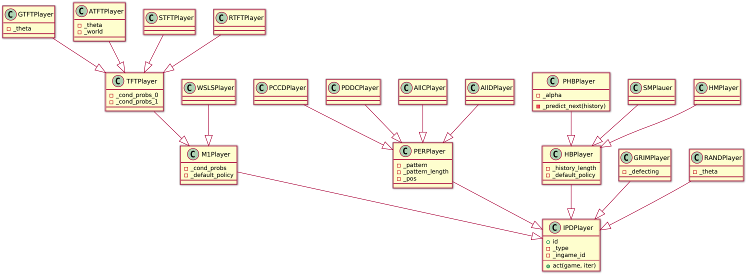


IPDEvolutionarySimulation
_player_types
_payoffs
_population
_n_generations
_n_iters
_noise
_average
create_fixed_generation(player_types, population_per_type)
create_generation(player_types, player_likelihoods, population)
plot_simulation(frame)
simulate(first_generation, n_generations, printing)
IPDPairwiseCompetition
_payoffs
_players
simulate(iters)
IPDPlayer
id
_type
_ingame_id
act(game, iter)
IPDGame
_payoffs
_players
actions
rewards
payoffs()
get_actions(start, end)
simulate(iters)
1..*
2..*
1..*
2..*
1..*
1..*
1
1..*
2
1..*

Simulating IDP Game



Players



GTFTPlayer
_theta
ATFTPlayer
_theta
_world
STFTPlayer
RTFTPlayer
TFTPlayer
_cond_probs_0
_cond_probs_1
WSLSPlayer
PCCDPlayer
PDDCPlayer
AllCPlayer
AllDPlayer
PHBPlayer
_alpha
_predict_next(history)
SMPlauer
HMPlayer
M1Player
_cond_probs
_default_policy
PERPlayer
_pattern
_pattern_length
_pos
HBPlayer
_history_length
_default_policy
GRIMPlayer
_defecting
RANDPlayer
_theta
IPDPlayer
id
_type
_ingame_id
act(game, iter)

Player types

Implementing player strategies

Fifteen different concrete agent behaviours have been implemented, for a total of ninenteen classes, structured on a generalization hierarchy whose root is **IPDPlayer**.

Aggregating by taking into account the ability of to exploit the information about the payoff matrix or the past actions:

Deterministic and/or periodic agents whose decisions are fixed or highly predicible. They do not take advantage of the history of previous rewards and actions

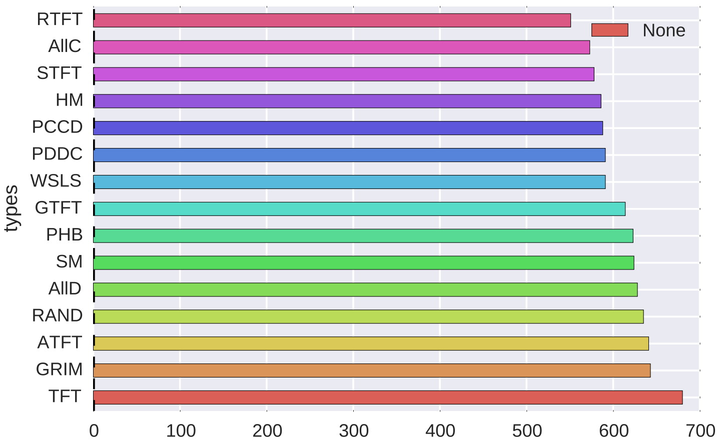
Memory-1 (stochastic) players that are influenced by some random variable or probability distribution. *Memory-N* agents derive these probabilities from the previous N moves (of both players)

History based their decision is highly dependent on the (partial or total) history of the opponent moves

Among the not implemented strategies, some noteworthy ones are: players **operating in groups**, or players **extorting payoffs**

Simulating

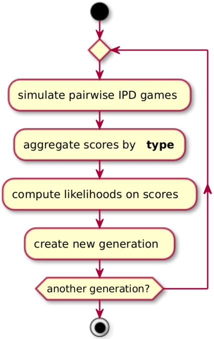
Exhaustive Pairwise IPD Simulation



Evolutionary game

Simulating Evolution

Start with a fixe sized population of L players extracted uniformly from K types.
Then, for H iterations (*generations*), play an exhaustive pairwise simulation on the current population and get each type score by aggregating agent scores.
Create a new generation by sampling new agents whose types are randomly chosen and are proportional to their normalized scores.



Simulation Results