Learning about the opponent’s accuracy and discounting

Andriy Burkov

Laval University, Canada
The main idea (while still criticizable) is to suppose that our opponent use the same reasoning, as we do, and has the same information and observability as we do.

However, it can differ from us by the values of the following two (unobservable) parameters:

- Discounting $0 \leq \delta \leq 1$, as usual exponentially weighing the today’s and tomorrow’s rewards.

- Accuracy, $c > 1/2$, measuring the probability to perfectly execute its intended action (can be viewed as $1$—noise in our observations of its actions).
It is also supposed that our opponent *does not* use the same learning algorithm. So, his $\delta$ and $c$ do not change with time, while ours (from his point of view) can change.

If we relax this assumption, than, in an unbounded case, to choose a strategy we will need to reason about how our change will change the strategy of the opponent and how then our strategy needs to be changed and how then the opponent’s strategy will be changed and how then we...

- Any multiagent learning algorithm must deal with such recursive reasoning. The assumption above permits avoiding it (some real applications can be found where it can be justified).

- However, in a general case, the criterion for the end of the recursion must me justified (cost of computing resources? memory bound? sort of discounting?)
Game and notation

- The game is the modified repeated Prisoner’s Dilemma as follows:

\[ M_t \cdot \begin{pmatrix} R & R \\ S & T \\ T & S \\ P & P \end{pmatrix} \]

- where \( M_t \) is a time step \( t \)'s instantiation of an unknown random variable (payoff multiplier).
- \( \bar{M} \) is \( M \)'s known average value.
- Players’ actions are \( C \) and \( D \), the game’s matrix is known, the instantiations of multipliers are observable before an action is taken.
- \( \hat{\pi}_C \) and \( \hat{\pi}_D \) are estimated payoffs for two possible strategies.
Decision rule

ACTION- FROM- PAYOFFS($\hat{\pi}_C, \hat{\pi}_D$)
1. if $\hat{\pi}_C \geq \hat{\pi}_D$
2. then return COOPERATE
3. else return DEFECT
Decision rule

Define,

\[ \hat{R} = \hat{c}_s \hat{c}_o R + \hat{c}_s (1 - \hat{c}_o) S + (1 - \hat{c}_s) \hat{c}_o T + (1 - \hat{c}_s)(1 - \hat{c}_o) P \]

\[ \hat{T} = \hat{c}_s \hat{c}_o T + \hat{c}_s (1 - \hat{c}_o) P + (1 - \hat{c}_s) \hat{c}_o R + (1 - \hat{c}_s)(1 - \hat{c}_o) S \]

\[ \hat{P} = \hat{c}_s \hat{c}_o P + \hat{c}_s (1 - \hat{c}_o) T + (1 - \hat{c}_s) \hat{c}_o S + (1 - \hat{c}_s)(1 - \hat{c}_o) R \]

Then,

\[ \hat{\pi}_C(\bar{M}, M_t, \hat{R}, \delta_s, \delta_o) = M_{t_0} \hat{R} + \bar{M} \hat{R} \frac{\min(\delta_o, \delta_s)}{1 - \min(\delta_o, \delta_s)} \]

\[ \hat{\pi}_D(\bar{M}, M_t, \hat{T}, \hat{P}, \delta_s, \delta_o) = M_t \hat{T} + \bar{M} \hat{P} \frac{\min(\delta_o, \delta_s)}{1 - \min(\delta_o, \delta_s)} \]
Modeling the opponent

- First, the finite set $\Theta$ of opponent types is created.
- Then, to each $\theta_{c,\delta} \in \Theta$ an initial probability is assigned.
- Before each play, a prediction of the opponent action for each opponent type is made as follows,

\[
\text{PREDICT- ACTION}(\theta)
\]

1. $\hat{\pi}_C \leftarrow \hat{\pi}_C(M_t, \tilde{M}, \tilde{R}, \delta[\theta], \delta[\theta])$
2. $\hat{\pi}_D \leftarrow \hat{\pi}_D(M_t, \tilde{M}, \hat{T}, \hat{P}, \delta[\theta], \delta[\theta]))$
3. $\text{return ACTION- FROM- PAYOFFS}(\hat{\pi}_C, \hat{\pi}_D)$

- Then, the opponent action is observed, and counters are updated:

\[
\text{LEARN- NAIVE- HARSANYI}(\text{learner, opponent})
\]

1. for $\theta \leftarrow \text{hypotheses[learner][opponent]}
2. do if $\text{action[opponent]} = \text{PREDICT- ACTION(\theta)}$
3. then $\text{INCREMENT(succes\[\theta])}$
4. else $\text{INCREMENT(failures[\theta])}$
Choosing own action

The following procedure is used to choose own actions given the belief over the opponent models:

```
CHOICE- ACTION- NAIVE- HARSANYI(player, opponent)
1   \( \hat{\pi}_C \leftarrow 0 \)
2   \( \hat{\pi}_D \leftarrow 0 \)
3   for \( \theta \leftarrow hypotheses[player][opponent] \)
4     do trials \leftarrow successes[\theta] + failures[\theta]
5     \quad P_{success} \leftarrow competence[\theta]
6     \quad P[\theta] \leftarrow \text{BINOMIAL}(successes[\theta], trials, P_{success})
7   \quad \pi_C[\theta] \leftarrow \hat{\pi}_C(M_t, \bar{M}, \hat{R}, \delta[player], \delta[\theta])
8   \quad \pi_D[\theta] \leftarrow \hat{\pi}_D(M_t, \bar{M}, \hat{T}, \hat{P}, \delta[player], \delta[\theta])
9   \hat{\pi}_C \leftarrow \hat{\pi}_C + (P[\theta] \ast \pi_C[\theta])
10  \hat{\pi}_D \leftarrow \hat{\pi}_D + (P[\theta] \ast \pi_D[\theta])
11  \text{return ACTION- FROM- PAYOFFS}(\hat{\pi}_C, \hat{\pi}_D)
```
Better classifying the opponent
Better classifying the opponent 2

\[ P[\theta] = \frac{\prod_{m \in M} \text{BINOMIAL}(\text{successes}[\theta][m], \text{trials}[\theta][m], c[\theta])}{\sum_{\theta \in \Theta} \prod_{m \in M} \text{BINOMIAL}(\text{successes}[\theta][m], \text{trials}[\theta][m], c[\theta])} \]