Artificial Intelligence and Collusion:

An Experiment

G. Calzolari, E. Calvano, V. Denicolò and S. Pastorello

EUI, University of Bologna and CEPR

EAGCP meeting, December 4th 2018

Algorithmic pricing

- Algorithmic pricing is here to stay
- Two vintages of software:
  1. estimate demand and form beliefs about competitors following the instructions of the programmer and then maximize profits
     - Can collude only to the extent they are designed or instructed to do so
  2. based on Artificial Intelligence, learn everything from scratch (machine learning)
     - All they need to be told is what variables to condition prices on (e.g., competitors' past prices)

Algorithmic pricing and collusion

- Is there hard evidence of algorithmic collusion?
  - Not yet (or very little)
- Are there reasons to be concerned?
  - Yes

Findings

- We follow an experimental approach
- We perform a number of computer simulations with machine learning algorithms that interact repeatedly over time in controlled economic environments
- We find that relatively simple pricing algorithms systematically learn to play sophisticated collusive strategies
    - Such strategies involve punishments that are proportional to the extent of the deviations and have a finite duration, with a gradual return to the pre-deviation prices

Tacit collusion

- The algorithms leave no trace of explicit collusion
  - The algorithms learn to play collusive strategies by trial and error, with no prior knowledge of the environment in which they operate
  - They have not been designed or instructed to collude
  - They do not communicate with each other

Findings

- We are the first to clearly document collusion among pricing algorithms
- Previous literature (in both computer science and economics) has sometimes found supra-competitive prices
- But high prices might be the result of the algorithms' failing to learn the static Nash equilibrium
    - If this is so, there would be little reason to be concerned as the problem would presumably fade away as the machine learning technology improves
    - If instead the algorithms learn to collude, then the problem might worsen with the diffusion of smarter programs

Findings

- Algorithms may in fact be better at colluding than humans
- The experimental literature has found that human subjects exhibit a limited ability to coordinate in the absence of explicit communication
- In the lab
  - two agents who cannot communicate with each other sometimes manage to converge to slightly supra-competitive prices
  - three agents typically set prices that are close to the static Nash equilibrium
  - four agents or more tend in fact to be more aggressive than Nash
    - This "rivalry" effect which is often attributed to the tendency of experimental subjects to behave as if they were involved in a contest
- With pricing algorithms, substantial collusion remains even with three or four active firms
  - even though the level of collusion does decrease with the number of competitors, as theory predicts

Tacit collusion

- In most countries today tacit collusion is not regarded as illegal
- The rationale for this policy is twofold
  - On the one hand, tacit collusion is viewed as illusory and very hard to achieve
    - Tolerant policy entails only few false negatives
  - On the other hand, tacit collusion is hard to detect
    - Aggressive policy implies many false positives
- The advent of algorithmic pricing may change the balance between type I and type II errors

Q-learning

- We focus on a class of Reinforcement Learning models known as Q-learning
- Several reasons for this, Q-learning is
    - Natural
    - Model free
    - Popular
    - Guaranteed to work is well behaved single decision making problems
    - Successful
        - Outperforms humans in complex games such as Go

Q-learning

- Q-Learning is designed to tackle Markov Decision Processes
- In each period $t = 0,1,2, \ldots$an agent observes a state variable $s_t \in S$ and then chooses an action $a_t \in A$
- The agent obtains a reward $\pi = F(a_t, s_t)$ and the system moves to the next state $s_{t+1} = G(a_t, s_t)$
- The objective is to maximize the present value of the reward stream

$$\sum_{t=0}^{\infty} \delta^t \pi(a_t, s_t)$$

- Here $\delta < 1$ represents the discount factor

Q-learning

- The algorithm tries to find the optimal policy without knowing the underlying model $\pi = F(a_t, s_t)$ and $s_{t+1} = G(a_t, s_t)$
- It does so by iteratively estimating the Q-function

$$Q(a,s) = \pi(a,s) + \delta \max_a [Q(a,s')]$$

- This is related to the value function by $V(s) \equiv \max_a Q(a,s)$
- With finite action and state spaces, the Q-function is a $|S| \times |A|$ matrix

Q-learning

- Q-learning algorithms estimate the Q-matrix by starting from a clean slate and updating the matrix as follows:
- For $(a, s) = (a_t, s_t)$

$$Q_{t+1}(a, s) = (1 - \alpha)Q_t(a, s) + \alpha \left[\pi(a, s) + \delta \max_a [Q_t(a, s')]\right]$$

- For $(a, s) \neq (a_t, s_t)$

$$Q_{t+1}(a, s) = Q_t(a, s)$$

- The variable α is the learning rate

Exploration

- To learn the optimal policy, the algorithm must explorate
- With ε-greedy exploration, the algorithm chooses the action currently perceived as optimal with probability ϵ and randomizes uniformly across all possible actions with probability $1 - \epsilon$
- **If ϵ is initially low but eventually goes to 1, then under mild technical conditions the algorithm converges to the optimal policy**

Q-learning in repeated games

- Q-learning algorithms may be also applied to infinitely repeated games
- To have a finite and time-invariant state space, one must assume bounded recall
    - The algorithms have a finite memory k, so they remember rivals' actions in the last k repetitions of the game
- Still, the environment faced by each agent is no longer stationary since the competing algorithms may change their behavior as they are experimenting and learning
- For this reason, there are no general convergence results for Q-learning algorithms in repeated games
    - We do not know whether the algorithms converge at all
    - If they do, we do not know whether they converge to an optimal strategy – hence a Nash equilibrium

State of the art

- The Q-learning algorithms we use are not state-of-the-art in machine learning
- We use the **independent learning** approach in which the algorithms do not realize they are playing a game
  - Extensive research on **joint learning** but no consensus yet
- Q-learning works for finite action and state spaces and becomes slower and slower as these spaces get bigger
  - **Deep learning** can deal with the continuous case
  - It has long suffered from problems of convergence
    - Recently, however, progress has been made
    - This allowed machine learning software to reach superhuman performances in playing Atari videogames or board games such as Go

Economic model

- An infinitely repeated Bertrand oligopoly game
- n firms, Logit demand and constant marginal costs c_i

$$q_i = \frac{e^{\frac{p_i a_i}{\mu}}}{\sum_{j=1}^{n} e^{\frac{p_j a_j}{\mu}} + e^{\frac{a_0}{\mu}}}$$

- As a robustness check, we have also considered the case of linear demand derived from Singh&Vives preferences

Exploration and initialization

- We use the ε-greedy model with a time declining exploration rate

$$\epsilon_t = 1 - e^{-\beta t}$$

- The initial matrix has been set in accordance to the fact that initially all agents randomize uniformly across all possible actions
  - Robustness checks:
    - Clean slate
    - Bertrand-Nash
    - Monopoly

Discretization and memory

- To get finite action and state spaces, we have discretized the model
- Prices can take m equally spaced values in the interval from $\xi\%$ below Bertrand to $\xi\%$ above monopoly
- In the baseline experiment, $m = 15$ and $\xi = 10\%$
- In the baseline experiment, one-period memory $(k = 1)$
- So we have 15 actions and 225 states; the Q-matrix has 3375 entries

Exploration and learning

- We have let α range in the entire unit interval $(0,1)$
- As for β, we have focused on the interval $(0,4 \times 10^{-5})$
- The upper bound of the interval implies that cells of the Q-matrix that correspond to prices that are systematically regarded as sub-optimal by the algorithms may be visited no more than 3-4 times
  - With even less exploration, learning would be difficult and outcomes might depend on the way the matrix is initialized

Profit gain

- We often use an index of competitiveness which is comparable across different experiments, i.e. the average profit gain

$$\Delta = \frac{\bar{\pi} - \pi^N}{\pi^M - \pi^N}$$

- A competitive outcome corresponds to $\Delta = 0$, a perfectly collusive one to $\Delta = 1$
- A similar index may also be defined for each individual firm

Baseline experiment

- $m = 15$
- $\xi = 10\%$
- $k = 1$
- $n = 2$
- $\delta = 0.95$
- $a_i = 2$
- $a_0 = 1$
- $c_i = 1$
- $\mu = \frac{1}{2}$

Convergence

- We presume that convergence is achieved if, for 25,000 repetitions in a row, for both algorithms, and for all states, the optimal action (i.e., the one with the highest Q-value) does not change
- For each experiment, we run 1,000 sessions which continue until convergence (but in any case for no more than one billion repetitions)
- While there is no theoretical guarantee that the algorithms converge, in more than 99.9% of our sessions they do
- Convergence may be slow. For example, with $\alpha = 1/2$ and $\beta = 2 \times 10^{-5}$ (the mid-points of our grid) convergence takes on average 500,000 periods

Time to convergence

- In fact, convergence is slow if the algorithms learn on-line
- But it is fast if they learn off-line (less than one minute of CPU time)
- Our approach is to give the algorithms all the time is needed to complete they learning
- Q-learning algorithms learn mechanically but stubbornly
- Newer algorithms are smarter
    - Deep learning algorithms can be much faster

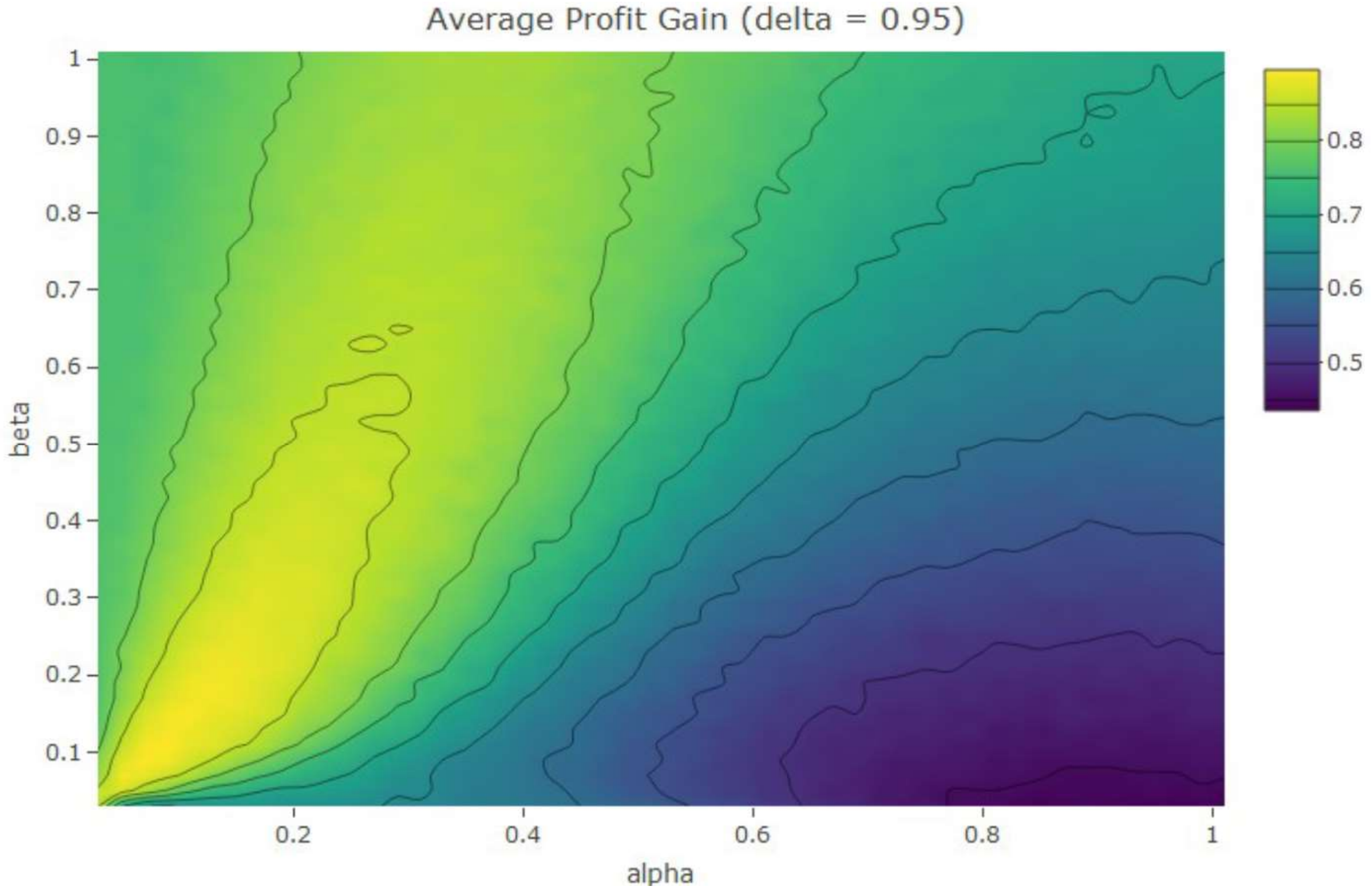
Consistency

- Even if learning requires considerable experimentation which creates a lot of noise, eventually algorithms do not make casual choices
- Long-run outcomes are quite stable across sessions
  - $\sigma_\Delta < 1\%$
  - In symmetric duopoly, $\Delta_1 - \Delta_2$ never statistically significant
- This stability of behavior does not depend on the fact that we average across the last 25,000 repetitions
- Upon convergence prices are quite stable
  - In more than 40% of the sessions, both algorithms keep charging the same price
  - The remaining sessions are characterized by price cycles. Of these cycles, however, more than three quarters have a period of two, and all involve adjacent prices
  - We interpret these cycles as an artifact of our discretization

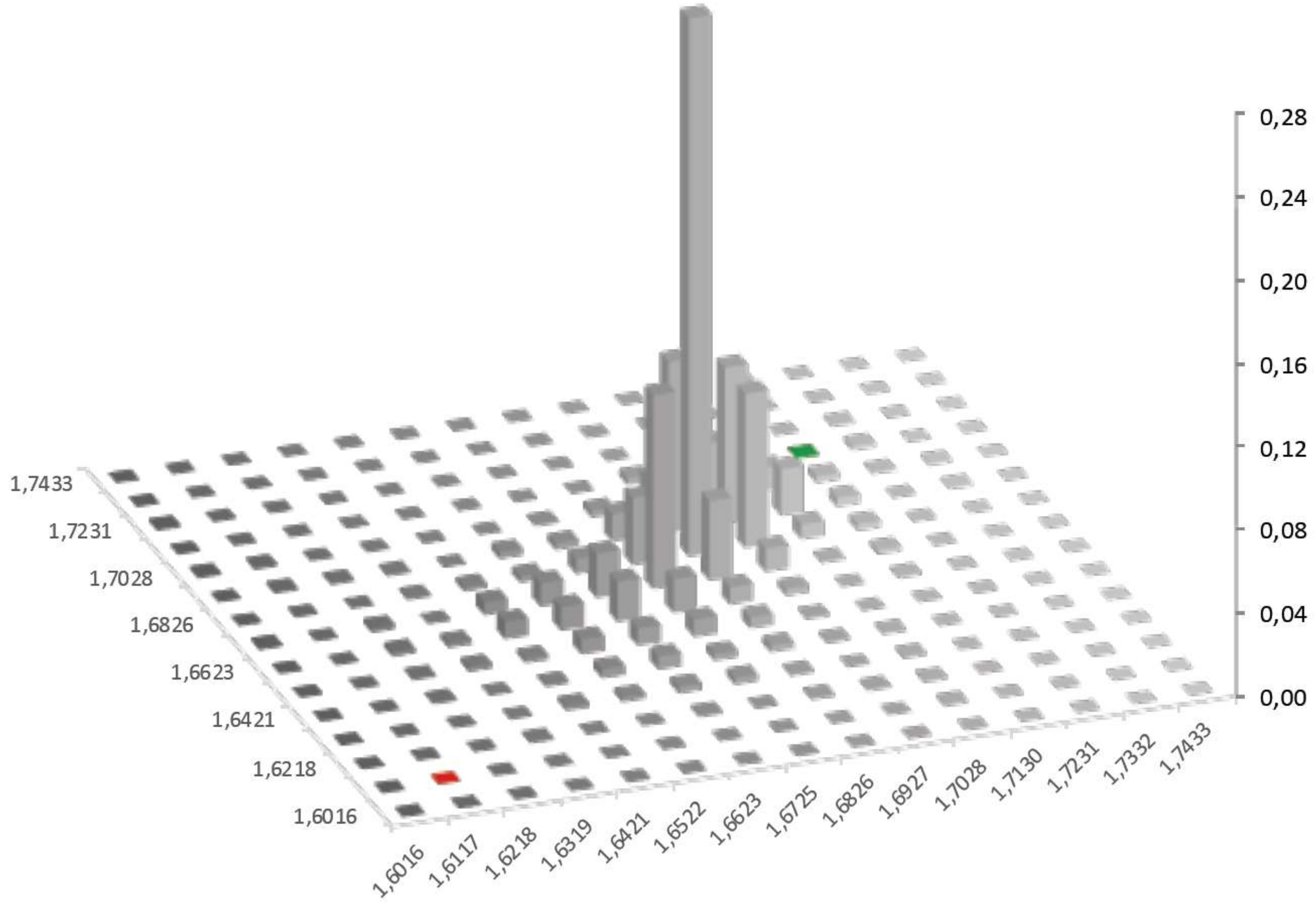
Equilibrium play

- Do algorithms learn an optimal strategy (i.e., a Nash equilibrium)?
- Again, no theoretical guarantee
- For relatively low values of α and β, we observe a substantial amount of equilibrium play on path
  - For example, when $\alpha = 0.1$ and $\beta = 8 \times 10^{-6}$ a Nash equilibrium is played 55% of the times, with each algorithm playing an individual best response more than 60% of the times
- When the algorithms do not play Nash, they play a strategy which is pretty close to a best response: the potential profit gain by playing a best response to the rival's strategy is around 1%
- Off path, things are quite different
  - A subgame perfect Nash equilibrium is reached in less than 2% of the sessions
- More equilibrium play is observed for even lower values of α and β, less when, for instance, α is large and β is small

Average profit gain



Prices



0,28
0,24
0,20
0,16
0,12
0,08
0,04
0,00
1,7433
1,7231
1,7028
1,6826
1,6623
1,6421
1,6218
1,6016
1,6016
1,6117
1,6218
1,6319
1,6421
1,6522
1,6623
1,6725
1,6826
1,6927
1,7028
1,7130
1,7231
1,7332
1,7433

Collusion?

- The key question is whether these high prices are the result of genuine collusion, or of the algorithms' failure to learn the static Nash equilibrium
- The policy implications would be radically different
  - if the algorithms end up charging high prices because they are not smart enough, the problem is likely to fade away as the technology improves
  - If they do learn to collude, the problem will presumably worsen as the programs become smarter

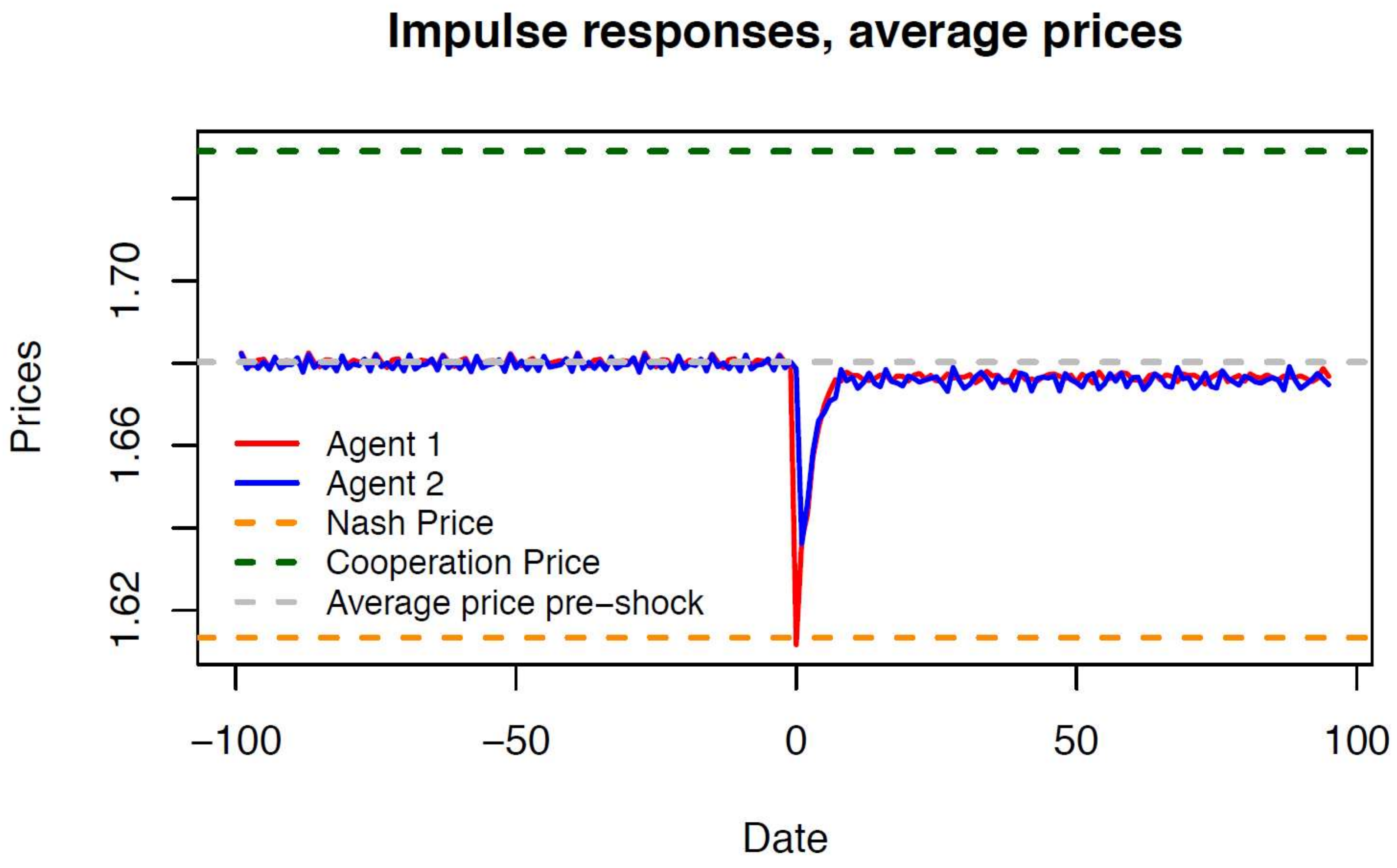
Test 1

- What do our algorithms learn when collusion cannot be an equilibrium phenomenon?
- Two cases:
  - $k = 0$ (no memory)
  - $\delta = 0$ (myopic behavior)
- In both cases, we find that the average profit gain is never greater than 10-20% and is often close to 0

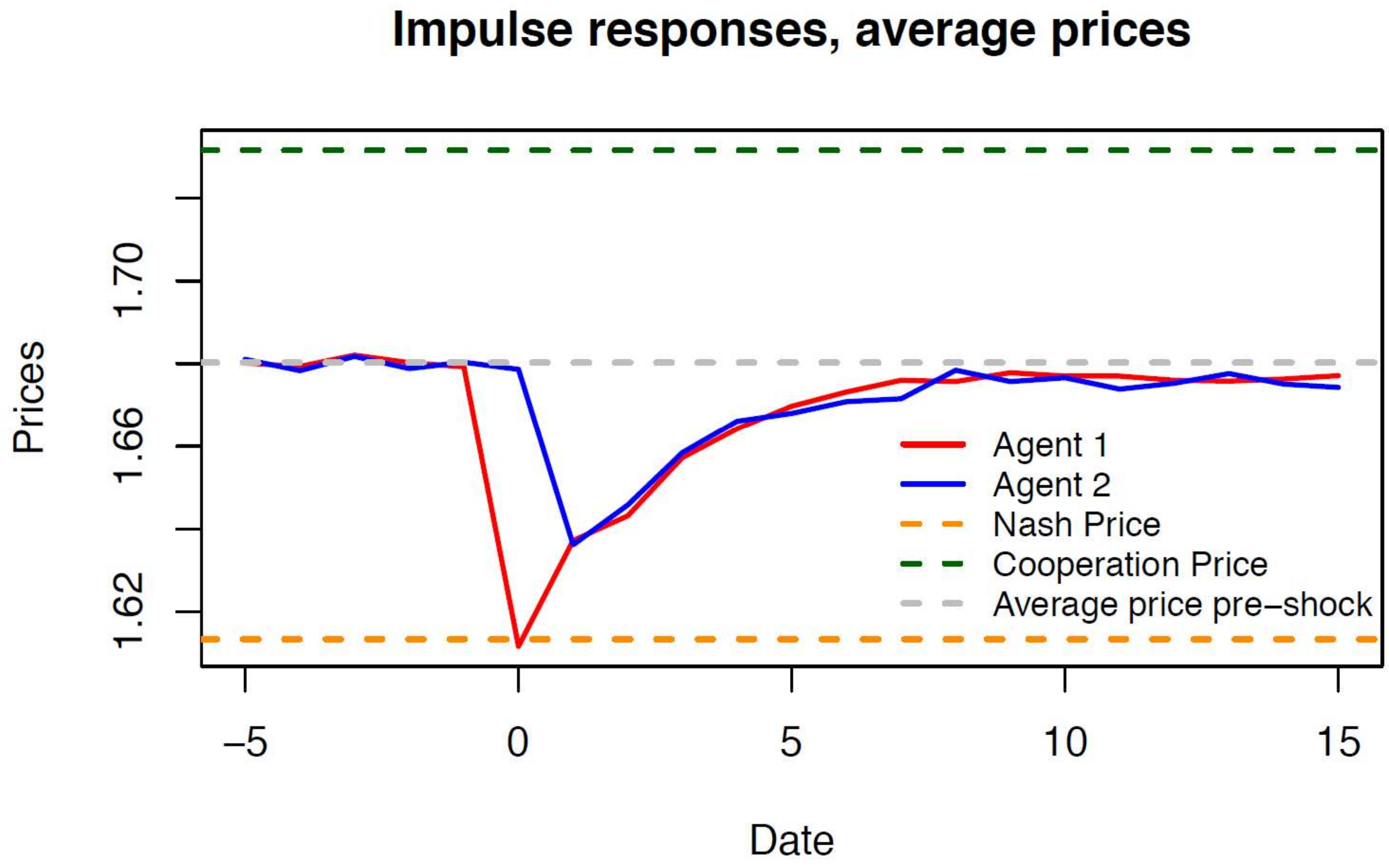
Test 2

- To understand the structure of the strategies that support cooperation, we perform the following exercise:
  - At the end of a session, we let the agents play for a number of periods according to the learnt strategies
  - Then we step in and manually override one agent's choice forcing him to choose the **static** best-response to the price that the opponent is playing on path
  - We then look at the reaction of the agents in the periods that follow
- In short, we derive "impulse-response" functions

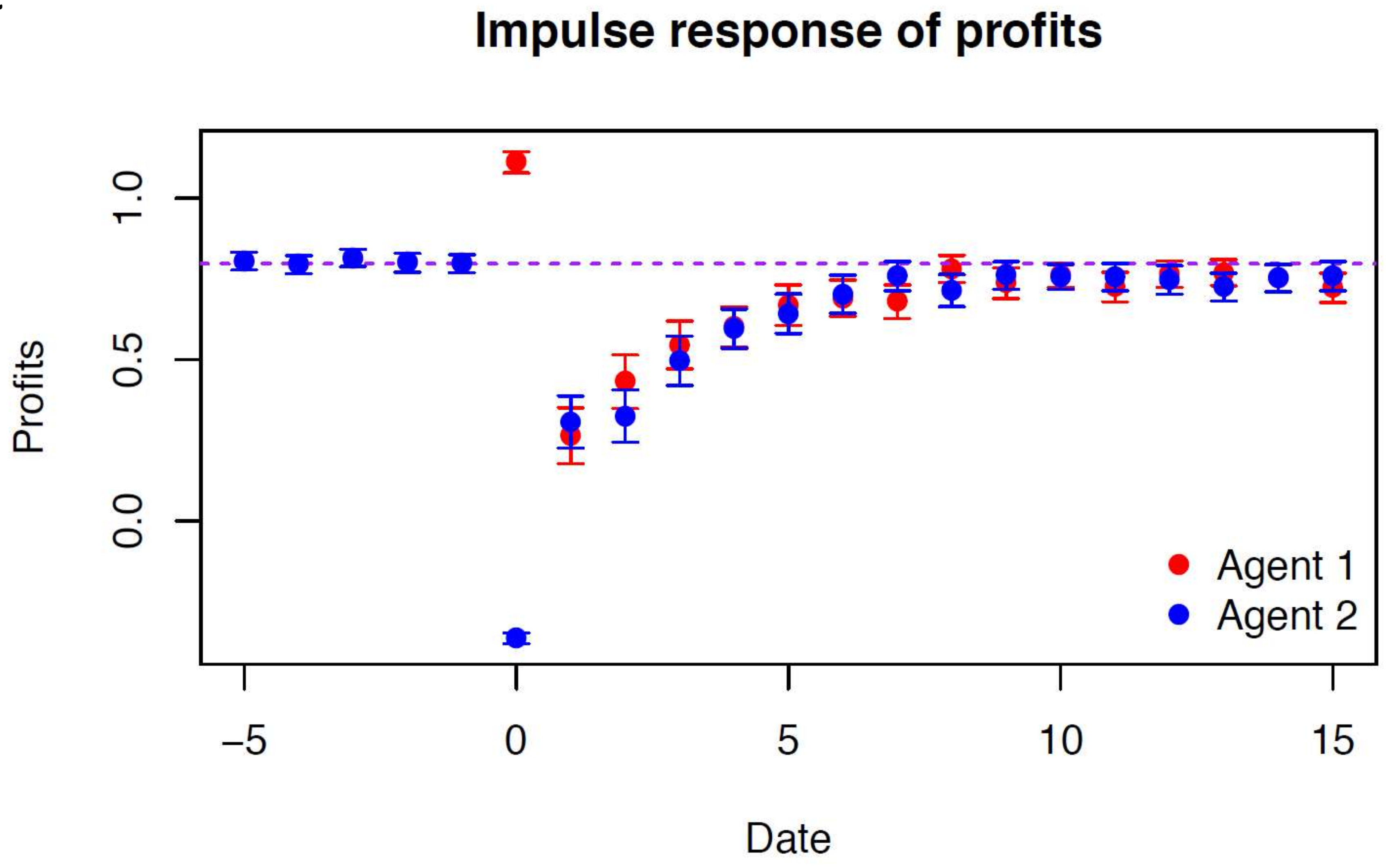
Impulse response



Zooming in

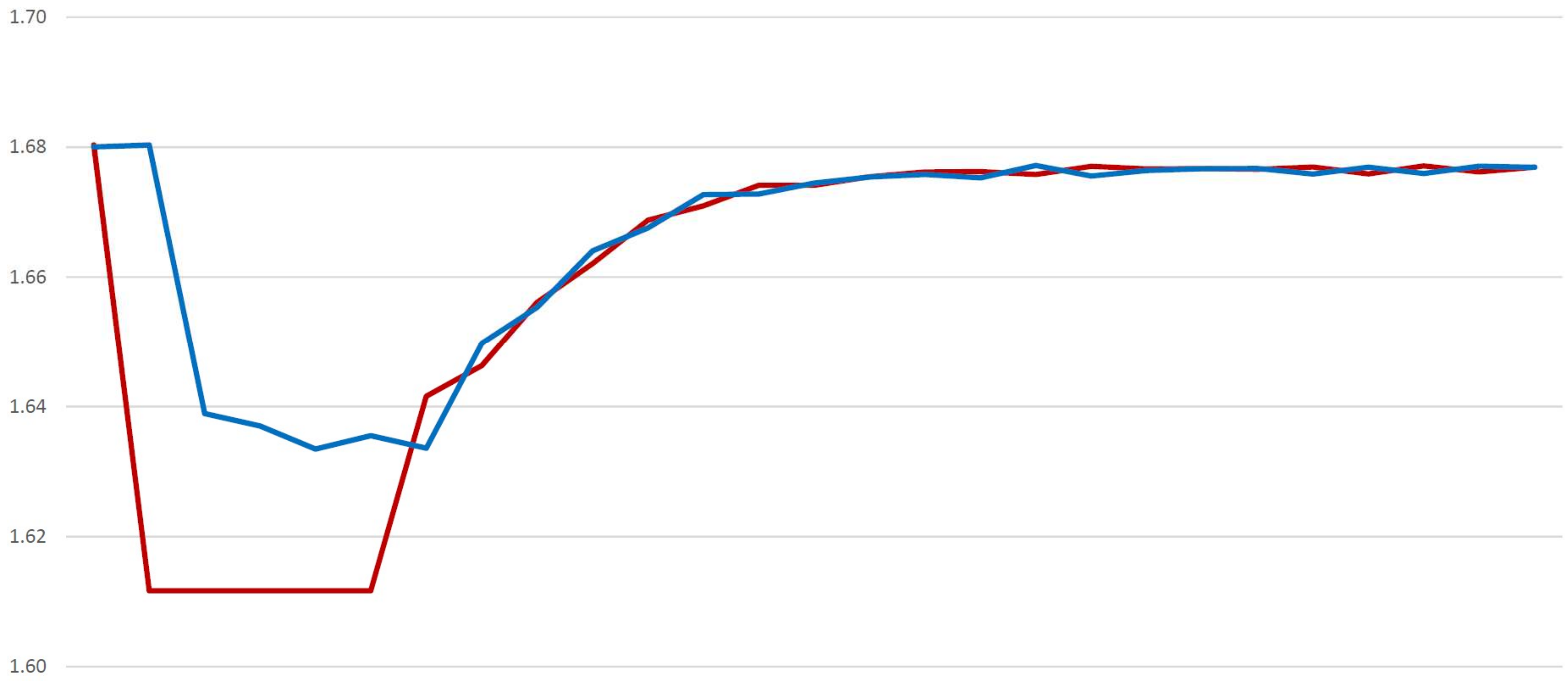

Impulse responses, average prices
Prices
1.62
1.66
1.70
Date
−5
0
5
10
15
Agent 1
Agent 2
Nash Price
Cooperation Price
Average price pre−shock

Profits



Impulse response of profits
Profits
1.0
0.5
0.0
Date
−5
0
5
10
15
● Agent 1
● Agent 2

5-period deviation



More firms

- In the lab, supra-competitive prices disappear as soon as there are three or more competing firms
- We have looked at the case $n = 3$ and $n = 4$
- For $\alpha = 0.05$ and $\beta = 8 \times 10^{-6}$, results are reported below

	$\boldsymbol{n = 2}$	$\boldsymbol{n = 3}$	$\boldsymbol{n = 4}$
Δ	80%	74%	70%

Asymmetric firms

- Collusion is notoriously more difficult when firms are asymmetric
- We have considered both the case of cost and demand asymmetries
- Results are similar
- With $c_1 = 1$ and $c_2 = 0.875$ (which implies a market share for the more efficient firm of 55%), for $\alpha = 0.05$ and $\beta = 8 \times 10^{-6}$ we have

	Symmetric	**Asymmetric**
Δ	80%	78%

Robustness

- Change in δ
- Asymmetric α and β
- Change in demand level
- Change in horizontal differentiation
- Stochastic demand
- Stochastic entry and exit
- More actions ($m = 30{,}50{,}100$)
- Longer memory ($k = 2$)
- Asyncrhonous learning

Open issues

- Non-stationary economic environments
- Algorithms learn in a different environment than the one where they operate
- Deep learning
- How do algorithms manage to coordinate so well? What are they comparative advantages over humans?