Autonomous Algorithmic Collusion: Q-Learning Under Sequential Pricing

Timo Klein
Tinbergen Institute; University of Amsterdam

Bergen Competition Policy Conference 2019
BECCLE, Norway

26 April 2019
1. Can AI-driven pricing algorithms learn to collude?
2. Would this be a competition law infringement?

Concerns mostly based on intuitive interpretation of AI
Many skeptical that this is even a problem
1. Can AI-driven pricing algorithms learn to collude?
2. Would this be a competition law infringement?
1. Can AI-driven pricing algorithms learn to collude?
2. Would this be a competition law infringement?

- Concerns mostly based on intuitive interpretation of AI
- Many skeptical that this is even a problem
Outline

1. Literature
2. Primer on Reinforcement Learning and Q-Learning
3. Environment and Algorithm
4. Simulation Results
5. Conclusions
1. Literature

2. Primer on Reinforcement Learning and Q-Learning

3. Environment and Algorithm

4. Simulation Results

5. Conclusions
Calvano, Calzolari, Denicolo and Pastorello (working paper, 2019)

- Also look at Q-learning collusion
- Results generally aligned
- Differences:
  1. Updates occur simultaneously instead of sequentially
  2. Allow for and require self-reactive conditioning (non-Markov)
  3. Explicit analysis of punishment strategies
Kuhn and Tadelis (2017), Schwalbe (2018)
- Humans and algorithms similarly ill-equipped to tacitly coordinate
- Would assume similar cognition for humans and AI

- Use forms of Q-learning in oligopoly environments
- Full knowledge; Not robust; Do not produce equilibrium behavior

Cooper et al. (2015)
- Certain revenue management convention may lead to collusion
- Not equilibrium behavior

Salcedo (2015)
- Collusion inevitable if short-run strategy commitments and 'decode'
- May be framed as communication; Conditions may not hold

Miklos-Thal and Tucker (2019)
- Better demand prediction may require lower cartel prices
1 Literature

2 Primer on Reinforcement Learning and Q-Learning

3 Environment and Algorithm

4 Simulation Results

5 Conclusions
Reinforcement Learning

Figure: Sutton and Barto (2018)
Reinforcement Learning

**Q-Learning** (Watkins, 1989)

- Popular and well-established type of reinforcement learning
- Aims to maximize sum of discounted rewards in unknown environment
- Strong theoretical properties in single-agent settings
Q-Learning

- $Q(a, s)$ estimates discounted rewards from action $a \in A$ in state $s \in S$
- Tabular case: $Q$ is a $|A| \times |S|$ matrix
Primer

Q-Learning

- \(Q(a, s)\) estimates discounted rewards from action \(a \in A\) in state \(s \in S\)
- Tabular case: \(Q\) is a \(|A| \times |S|\) matrix

Learning Module

- Take \(s\) as old state and \(s'\) as new state
- Recursive updating:
  \[
  Q(a, s) \leftarrow (1 - \alpha)Q(a, s) + \alpha \left( R(a, s, s') + \delta \max_{a'} Q(a', s') \right)
  \]

Action-Selection Module

- Exogenously programmed to trade off exploitation-exploration
Primer

Q-Learning

- \( Q(a, s) \) estimates discounted rewards from action \( a \in A \) in state \( s \in S \)
- Tabular case: \( Q \) is a \( |A| \times |S| \) matrix

Learning Module

- Take \( s \) as old state and \( s' \) as new state
- Recursive updating:
  \[
  Q(a, s) \leftarrow (1 - \alpha)Q(a, s) + \alpha \left( R(a, s, s') + \delta \max_a Q(a, s') \right)
  \]

Action-Selection Module

- Exogenously programmed to trade off exploitation-exploration
- Provably converges to optimal policy under single-agent learning
- No theoretical guarantee under multi-agent learning
Environment and Algorithm

1 Literature

2 Primer on Reinforcement Learning and Q-Learning

3 Environment and Algorithm

4 Simulation Results

5 Conclusions
Environment: Sequential Competition

- Maskin and Tirole (1988), firms $i \in \{1, 2\}$ set prices in turn
Environment and Algorithm

Environment: Sequential Competition

- Maskin and Tirole (1988), firms $i \in \{1, 2\}$ set prices in turn
- Prices $p^i \in \{0, \frac{1}{k}, \frac{2}{k}, \ldots, 1\}$, so $k$ intervals between 0 and 1
Environment: Sequential Competition

- Maskin and Tirole (1988), firms $i \in \{1, 2\}$ set prices in turn
- Prices $p^i \in \{0, \frac{1}{k}, \frac{2}{k}, \ldots, 1\}$, so $k$ intervals between 0 and 1
- Per-period profit $\pi^i = (p^i - c^i)q^i$
Environment and Algorithm

Environment: Sequential Competition

- Maskin and Tirole (1988), firms $i \in \{1, 2\}$ set prices in turn
- Prices $p^i \in \{0, \frac{1}{k}, \frac{2}{k}, \ldots, 1\}$, so $k$ intervals between 0 and 1
- Per-period profit $\pi^i = (p^i - c^i)q^i$
- Objective $\max \sum_{s=0}^{\infty} \delta^s \pi_{t+s}^i$
Environment: Sequential Competition

- Maskin and Tirole (1988), firms $i \in \{1, 2\}$ set prices in turn
- Prices $p^i \in \{0, \frac{1}{k}, \frac{2}{k}, \ldots, 1\}$, so $k$ intervals between 0 and 1
- Per-period profit $\pi^i = (p^i - c^i)q^i$
- Objective $\max \sum_{s=0}^{\infty} \delta^s \pi^i_{t+s}$
- Scope: homogeneous good, linear demand, 2 firms

$$q^i = \begin{cases} 
1 - p^i & \text{if } p^i < p^j \\
0.5(1 - p^i) & \text{if } p^i = p^j \\
0 & \text{if } p^i > p^j 
\end{cases}$$
Environment and Algorithm

Algorithm: Sequential Q-Learning

Learning Module

- Take old state $s = p^j_{t-1}$ and new state $s' = p^j_{t+1}$,

$$Q(p^i_t, s) \leftarrow (1 - \alpha)Q(p^i_t, s) + \alpha \left( \pi(p^i_t, s) + \delta \pi(p^i_t, s') + \delta^2 \max_p Q(p, s') \right)$$

Action-Selection Module

- Explores with probability $\varepsilon_t$ ⇒ Pick any $p$
- Exploits with probability $1 - \varepsilon_t$ ⇒ Pick $p$ that maximizes $Q(p, s)$

Still very basic algorithm:
1. Slow and inefficient learning
2. Untargetted exploration
Algorithm: Sequential Q-Learning

Learning Module

- Take old state \( s = p^j_{t-1} \) and new state \( s' = p^j_{t+1} \),

\[
Q(p^i_t, s) \leftarrow (1 - \alpha)Q(p^i_t, s) + \alpha \left( \pi(p^i_t, s) + \delta \pi(p^i_t, s') + \delta^2 \max_p Q(p, s') \right)
\]

Action-Selection Module

- Explores with probability \( \varepsilon_t \) \( \Rightarrow \) Pick any \( p \)
- Exploits with probability \( 1 - \varepsilon_t \) \( \Rightarrow \) Pick \( p \) that maximizes \( Q(p, s) \)
Algorithm: Sequential Q-Learning

Learning Module

- Take old state $s = p^j_{t-1}$ and new state $s' = p^j_{t+1}$,

$$Q(p^j_{t}, s) \leftarrow (1 - \alpha)Q(p^j_{t}, s) + \alpha \left( \pi(p^j_{t}, s) + \delta \pi(p^j_{t}, s') + \delta^2 \max_{p} Q(p, s') \right)$$

Action-Selection Module

- Explores with probability $\varepsilon_t$ $\Rightarrow$ Pick any $p$
- Exploits with probability $1 - \varepsilon_t$ $\Rightarrow$ Pick $p$ that maximizes $Q(p, s)$

Still very basic algorithm:

1. Slow and inefficient learning
2. Untargetted exploration
(1) **Profitability:** \( \Delta^i = \frac{\text{Expected profit gains}}{\text{Joint-profit maximizing gains}} = \frac{Q^i(p^i, s) - Q^N}{Q^C - Q^N} \)

- \( \Delta^i = 1 \) joint-profit maximizing outcome
- \( \Delta^i = 0 \) competitive outcome (defined as static Nash)
Performance Metrics

(1) Profitability: \[ \Delta^i = \frac{\text{Expected profit gains}}{\text{Joint-profit maximizing gains}} = \frac{Q^i(p^i, s) - Q^N}{Q^C - Q^N} \]

- \( \Delta^i = 1 \) joint-profit maximizing outcome
- \( \Delta^i = 0 \) competitive outcome (defined as static Nash)

(2) Optimality: \[ \Gamma^i = \frac{\text{Expected future profits}}{\text{Best-response future profits}} = \frac{Q^i(p^i, s)}{\max_p Q^i^*(p, s)} \]

- \( Q^i^* \) are the optimal Q-values given current competitor strategy
- \( \Gamma^i = 1 \) best response
- \( \Gamma^i < 1 \) shows degree below best response
Simulation Results

1. Literature
2. Primer on Reinforcement Learning and Q-Learning
3. Environment and Algorithm
4. Simulation Results
5. Conclusions
Simulation Results

Two cases:

1. Q-learning versus fixed-strategy tit-for-tat
2. Q-learning versus Q-learning

Simulation set-up:
Simulation Results

Two cases:

1. Q-learning versus fixed-strategy tit-for-tat
2. Q-learning versus Q-learning

Simulation set-up:

- Price set: $k = \{6, 12, 50\}$ possible prices
Two cases:

1. Q-learning versus fixed-strategy tit-for-tat
2. Q-learning versus Q-learning

Simulation set-up:

- Price set: $k = \{6, 12, 50\}$ possible prices
- $R = 1000$ runs of $T = 300(k + 1)^2$ periods each
Simulation Results

Two cases:

- (1) Q-learning versus fixed-strategy tit-for-tat
- (2) Q-learning versus Q-learning

Simulation set-up:

- Price set: \( k = \{6, 12, 50\} \) possible prices
- \( R = 1000 \) runs of \( T = 300(k + 1)^2 \) periods each
- Learning parameters: \( \alpha = 0.5, \delta = 0.95 \) and \( \varepsilon_t = (1 - \theta)^t \)
Simulation Results

Two cases:

(1) Q-learning versus fixed-strategy tit-for-tat
(2) Q-learning versus Q-learning

Simulation set-up:

Price set: $k = \{6, 12, 50\}$ possible prices

$R = 1000$ runs of $T = 300(k + 1)^2$ periods each

Learning parameters: $\alpha = 0.5$, $\delta = 0.95$ and $\varepsilon_t = (1 - \theta)^t$

$\theta$ such that $\varepsilon_t = 0.5\%$ halfway and $\varepsilon_t = 0.0025\%$ at the end
Simulation Results

Two cases:

- (1) Q-learning versus fixed-strategy tit-for-tat
- (2) Q-learning versus Q-learning

Simulation set-up:

- Price set: \( k = \{6, 12, 50\} \) possible prices
- \( R = 1000 \) runs of \( T = 300(k + 1)^2 \) periods each
- Learning parameters: \( \alpha = 0.5, \delta = 0.95 \) and \( \varepsilon_t = (1 - \theta)^t \)
- \( \theta \) such that \( \varepsilon_t = 0.5\% \) halfway and \( \varepsilon_t = 0.0025\% \) at the end
- Initiate \( Q \) with discounted perpetuity static Nash (not necessary)
(1) Q-learning versus fixed-strategy tit-for-tat, $k = 6$
Simulation Results

(2) Q-learning versus Q-learning, $k = 6$

**Conclusions**

- Profitability high
- $-194$ with both $\Delta = 1$
- Often optimal
- $-164$ with both $\Gamma = 1$
- $-63$ with $\Delta = \Gamma = 1$
Simulation Results

(2) Q-learning versus Q-learning, $k = 12$

Conclusions

- More profitable
- 34 with both $\Delta = 1$
- Often optimal for only one firm
- 27 with both $\Gamma = 1$
- 6 with $\Delta = \Gamma = 1$
Simulation Results

(2) Q-learning versus Q-learning, $k = 50$

Conclusions

- Even more profitable
- Less optimal
- 0 with $\Delta = \Gamma = 1$
- Price dynamics?
Adopts a fixed price or asymmetric price cycles

More asymmetric price cycles if $k$ is larger

<table>
<thead>
<tr>
<th></th>
<th>$k = 6$</th>
<th>$k = 12$</th>
<th>$k = 50$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runs with a fixed price</td>
<td>508/1,000</td>
<td>111/1,000</td>
<td>11/1,000</td>
</tr>
<tr>
<td>Runs with monopoly fixed price</td>
<td>194/1,000</td>
<td>35/1,000</td>
<td>0/1,000</td>
</tr>
<tr>
<td>Runs without a fixed price</td>
<td>492/1,000</td>
<td>889/1,000</td>
<td>989/1,000</td>
</tr>
<tr>
<td>Periods with a price decrease</td>
<td>47%</td>
<td>63%</td>
<td>76%</td>
</tr>
<tr>
<td>Periods with a price increase</td>
<td>22%</td>
<td>17%</td>
<td>11%</td>
</tr>
</tbody>
</table>

Table 1: Price dynamics final 100 periods
Market price dynamics final 40 periods, 3 random runs, $k = 50$
Jumps before reaching lower bound, to price above monopoly
Conclusions

1 Literature

2 Primer on Reinforcement Learning and Q-Learning

3 Environment and Algorithm

4 Simulation Results

5 Conclusions
Conclusions

- Autonomous algorithmic collusion in principle possible

Outcomes resemble equilibrium behavior ...

... but scope for more advanced algorithms

1. to guarantee optimality
2. to deal with less stylized environments

Many exciting areas for future research!
Conclusions

- Autonomous algorithmic collusion in principle possible
- Sequential Q leads to higher prices, only programmed to max own profits
Conclusions

- Autonomous algorithmic collusion in principle possible
- Sequential Q leads to higher prices, only programmed to max own profits
- Outcomes resemble equilibrium behavior ...

Multi-Agent Reinforcement Learning ⇒ see appendix
Deep Reinforcement Learning
Supervised Learning (function approximation)
Conclusions

- Autonomous algorithmic collusion in principle possible
- Sequential Q leads to higher prices, only programmed to max own profits
- Outcomes resemble equilibrium behavior ...
- ... but scope for more advanced algorithms
  1. to guarantee optimality
  2. to deal with less stylized environments

- Many exciting areas for future research!
  - Multi-Agent Reinforcement Learning ⇒ see appendix
  - Deep Reinforcement Learning
  - Supervised Learning (function approximation)
  - (...)

Mult-Agent Reinforcement Learning ⇒ see appendix