

Algorithmic Collusion Detection

Matteo Courthoud¹

September 16, 2021

¹University of Zürich, email: matteo.courthoud@econ.uzh.ch.

Algorithmic Collusion

- High-frequency pricing decisions are more and more often delegated to algorithms (Chen et al., 2016)
- Computational evidence that algorithms can learn to collude (Calvano et al., 2020; Klein, 2019)
- Empirical evidence that it might be happening (Assad et al., 2020)
- General concerns from policy: OECD (2017), Autorité de la concurrence & Bundeskartellamt (2019), UK Competition and Markets Authority (2021)

How can we detect collusion? We first need to define it.

Defining Algorithmic (Tacit) Collusion

From the **legal perspective**:

“Collusion is when firms use strategies that embody a reward–punishment scheme which rewards a firm for abiding by the supracompetitive outcome and punishes it for departing from it.” (Harrington, 2018)

- Normally we cannot observe firm strategies so **we punish firm intent** to collude, via communication
- However, we can observe algorithm strategies!

What is the best way to detect algorithmic collusion?

Detecting Algorithmic Collusion

Ex-ante

1. Simulate algorithm behavior to classify algorithms into collusive and non-collusive ones (Ezrachi & Stucke, 2017)

Live monitoring

2. Get intermediate output from the algorithm
 - ▶ *“Computer scientists might develop algorithms that explain their own behavior”* (Calvano et al., 2020)

Ex-post

3. Rely on observational data
4. Look at algorithm strategies
 - ▶ *“An AA [artificial agent] is not collusive if its price recommendation is not dependent on rival firms’ responding in a particular manner”* (Harrington, 2018)

This Project

1. Show that observing algorithms' strategies is not enough to detect collusion
 - ▶ Even algorithms' strategies that do not depend on rivals' actions can generate collusive behavior in the form of reward-punishment schemes
 - ▶ How? **Self-punishment**

This Project

1. Show that observing algorithms' strategies is not enough to detect collusion
 - ▶ Even algorithms' strategies that do not depend on rivals' actions can generate collusive behavior in the form of reward-punishment schemes
 - ▶ How? **Self-punishment**
2. Propose a method to detect collusive behavior
 - ▶ **Intuition:** q-learning algorithms are not built to learn SP strategies
 - ▶ Potential for exploiting them, when colluding
 - ▶ No modeling assumptions
 - ▶ Consistent **but** unfeasible: need control over all algorithms

This Project

1. Show that observing algorithms' strategies is not enough to detect collusion
 - ▶ Even algorithms' strategies that do not depend on rivals' actions can generate collusive behavior in the form of reward-punishment schemes
 - ▶ How? **Self-punishment**
2. Propose a method to detect collusive behavior
 - ▶ **Intuition:** q-learning algorithms are not built to learn SP strategies
 - ▶ Potential for exploiting them, when colluding
 - ▶ No modeling assumptions
 - ▶ Consistent **but** unfeasible: need control over all algorithms
3. Propose feasible implementation of (2)
 - ▶ Retrain algorithms unilaterally on historical data
 - ▶ **Outcome:** if (and only if) algorithms are colluding, they learn profitable deviations

Literature

- Q-learning algorithms learn grim-trigger strategies
[Calvano et al. (2020), Klein (2019)]
- Empirical evidence of the use of pricing algorithms
[Chen et al. (2016)]
- Empirical evidence of colluding pricing algorithms
[Assad et al. (2020)]
- Economic modeling can help preventing collusion
[Asker et al. (2021)]
- Legal perspective
[Mehra (2015), Ballard & Naik (2017), Capobianco & Gonzaga (2017), Ezrachi & Stucke (2017), Deng (2017), Gal (2017), Okuliar & Kamenir (2017), Harrington (2018), Calvano et al. (2019)]

Q-Learning

Q-Learning: Overview

Q-learning algorithms are model-free algorithms designed to find optimal policies in dynamic environments.

- Not intended to be deployed in strategic environments
- Most optimality results on Q-learning, derived for stationary environments

Setting

- Objective: maximize flow of discounted payoffs $\mathbb{E} [\sum_{t=0}^{\infty} \delta^t \pi_t]$
- In each period, an agent observes a state $s \in S$ (finite)
- The agent has to take an action $a \in A$ (finite)

Q-Learning: Overview

What does **model-free** means?

The algorithm does not rely on assumptions nor tries to model

- The relationship between states, actions, and payoffs $\pi(s, a)$
- The relationship between states, actions, and future states $s'(s, a)$

Feedback:

- The algorithm knows its own state and actions
- At the end of each period, it observes the realized payoffs and the next state

Q-Learning Algorithm

Timing

- Action-specific value function $Q(s, a)$ is initialized
- In each period
 1. The algorithm observes the state s
 2. Two different ways of selecting an action a
 - ▶ **Exploration:** select a random action $a \in A$
 - ▶ **Exploitation:** select the perceived best action $a = \arg \max_a Q(s, a)$
 3. The payoff $\pi(s, a)$ and next state $s'(s, a)$ realize
 4. The algorithm updates $Q(s, a)$ using $\pi(s, a)$ and $s'(s, a)$
 - ▶ Weighted average of observed payoffs in state s , given action a
- Convergence when actions are not updated for enough periods

Exploration vs Exploitation

- Trade-off
 - ▶ Exploration needed to learn policies' value
 - ▶ Exploitation to rip the benefits of the optimal policy
- Over time the algorithm shifts from “exploration-most” mode to “exploitation-only” mode
 - ▶ For local optimality in markov environments (Sutton & Barto, 2018)
- Note that the algorithm “learns” also in exploitation mode
 - ▶ It still updates Q
 - ▶ But only for optimal actions, in visited states
- A new action a'' can become optimal in a state s only if the previous optimal action a' performs worse than a'' for a sufficient number of visits of s
 - ▶ **Important:** in exploitation mode, action optimality is determined by the One Shot Deviation principle

Algorithm Details

Algorithm 1: Q-learning

initialize $Q_i^0(\mathbf{s}, a_i) \forall i = 1 \dots n, \mathbf{s} \in \mathcal{S}, a_i \in \mathcal{A}$;

initialize \mathbf{s}^0 ;

while *convergence condition not met* **do**

for $i = 1 \dots n$ **do**

 exploration _{i} = $\mathbb{I}(r_i < e^{-\beta t})$ where $r_i \sim U(0, 1)$;

if exploration _{i} **then**

$a_i^* = a \in A$ chosen uniformly at random ;

else

$a_i^* = \arg \max_{a_i} Q_i(\mathbf{s}, a_i)$;

end

end

 observe π given (\mathbf{s}, a^*) ;

 observe \mathbf{s}' given (\mathbf{s}, a^*) ;

$Q_i(\mathbf{s}, a_i^*) = \alpha Q_i(\mathbf{s}, a_i^*) + (1 - \alpha) [\pi(\mathbf{s}, a^*) + \delta \max_{a'_i} Q_i(\mathbf{s}', a'_i)] \quad \forall i$;

$\mathbf{s} = \mathbf{s}'$;

end

Experience Based Equilibrium

What does Q converge to?

Experience-based Equilibrium (Fershtman & Pakes, 2012): at the states of the equilibrium recurrent class

- strategies are optimal given agents' evaluations, and
- these evaluations are consistent with the empirical distribution of outcomes and the primitives of the model

Importantly, \neq from Subgame Perfect Equilibrium

Simulations

Setting

- Same as Calvano et al. (2020)
- Infinitely repeated game
- Unit mass of consumers with logit demand

$$q_i(\mathbf{p}, r) = \frac{e^{-\mu p_i}}{e^{-\mu p_i} + e^{-\mu p_j} + e^{-\mu 0}}$$

- Two firms compete in prices
- Fixed grid of available actions (prices)
 - ▶ Spans and includes both competitive and collusive prices
- State
 - ▶ Calvano et al. (2020): prices of both firms in the previous period $s = \mathbf{p}_{t-1} = (p_{i,t-1}, p_{-i,t-1})$
 - ▶ **Blind setting**: own price in the previous period $s = p_{i,t-1}$,

Calvano et al. (2020)

Firms strategies converge to supra-competitive prices.



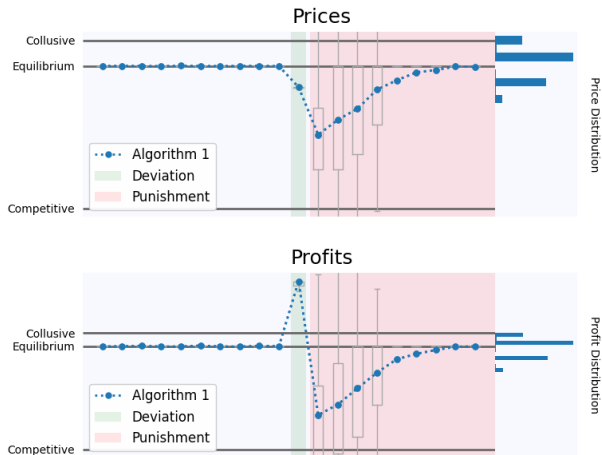
Calvano et al. (2020)

If one firm unilaterally deviates, it gets higher profits.



Calvano et al. (2020)

But the deviating firm gets punished afterwards.



Comment

How can we prevent collusion?

- Harrington (2018), Ezrachi & Stucke (2017): sufficient to have **policy not depend on competitors' actions**
- Why? It's sufficient to look at the algorithm's inputs to determine the risk of collusion.
 - ▶ At least in the stylized environment of Calvano et al. (2020)

What if the algorithm strategy was based only on own past action?

- Firms condition their strategies only on their own past actions
- Learn from competitor's action indirectly through payoffs
- However, policies Q do not depend on rival's action

Blind Algorithms

Same setting as Calvano et al. (2020), but algorithms observe only own past action.



Outcome: firms set supra-competitive prices. *How?* Self-punishment.

Blind Algorithms

The rival does not observe the deviation so does not change actions.



Why is it an equilibrium? How do algorithms learn these strategies?

Discussion

Why is it an equilibrium?

- Algorithm in exploitation mode only tests one-shot deviations
- Unable to learn completely new strategies

How do algorithms learn these strategies?

- In the exploration phase, the algorithm tests any strategy
- High-prices + self punishment is more profitable than NE

Is it collusive behavior?

- It is a reward-punishing scheme with the sole intent of keeping supracompetitive prices

Detecting Collusion

Detecting Collusion

1. Inspecting and understanding the algorithm might be hard
2. As in previous section, even when algorithms do not observe competitors' strategies, they can learn to collude
3. Detecting collusion from observational data is hard
 - ▶ *“Absent a natural experiment or counterfactual, enforcers may not readily discern whether (and why) the market price is too high. Is it the result of artificial intervention or natural supply and demand dynamics?”* (Ezrachi & Stucke, 2019)

Detecting Collusion

1. Inspecting and understanding the algorithm might be hard
2. As in previous section, even when algorithms do not observe competitors' strategies, they can learn to collude
3. Detecting collusion from observational data is hard
 - ▶ *“Absent a natural experiment or counterfactual, enforcers may not readily discern whether (and why) the market price is too high. Is it the result of artificial intervention or natural supply and demand dynamics?”* (Ezrachi & Stucke, 2019)

Idea: re-train algorithms unilaterally

- Intuition: if algorithms are able to learn to collude, they should be able to learn to exploit collusion.

Re-training Algorithms

Procedure

- Select one of the algorithms
- Freeze other algorithms (fix strategies)
- Reset the exploration/exploitation counter of the selected algorithm
- Iterate until convergence

Is it able to learn a new and more profitable strategy?

- Not trivial: if opponent's strategy was really subgame-perfect, no

Re-training Algorithms

Procedure

- Select one of the algorithms
- Freeze other algorithms (fix strategies)
- Reset the exploration/exploitation counter of the selected algorithm
- Iterate until convergence

Is it able to learn a new and more profitable strategy?

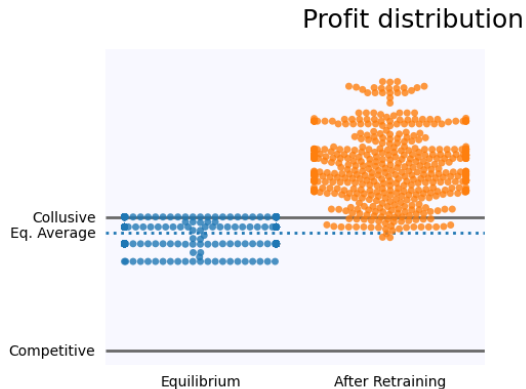
- Not trivial: if opponent's strategy was really subgame-perfect, no

I analyze three cases

1. Replication of Calvano et al. (2020)
2. Blind algorithms → collusive
3. Q-learning algorithm vs firm playing static NE (placebo)

Re-training

1. Replication of Calvano et al. (2020).

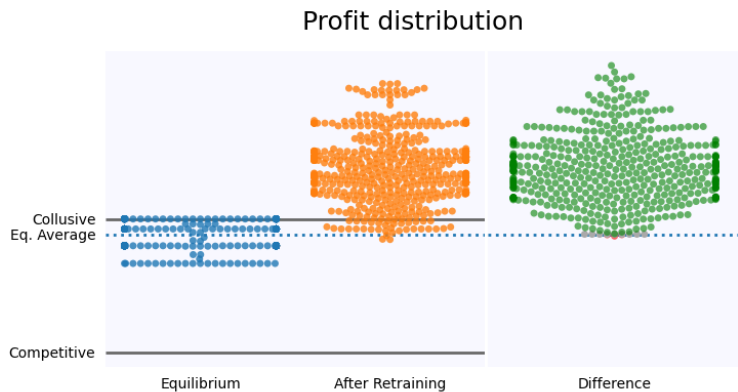


▶ prices

Outcome: consistent pattern of profitable deviations.

Re-training

1. Replication of Calvano et al. (2020).

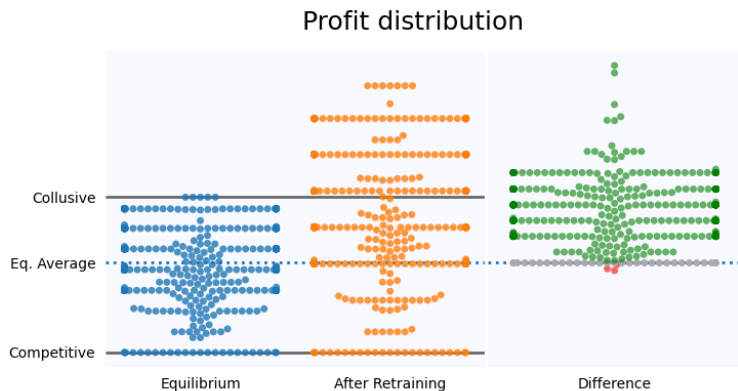


▶ prices

Outcome: consistent pattern of profitable deviations.

Re-training

2. Blind algorithms.

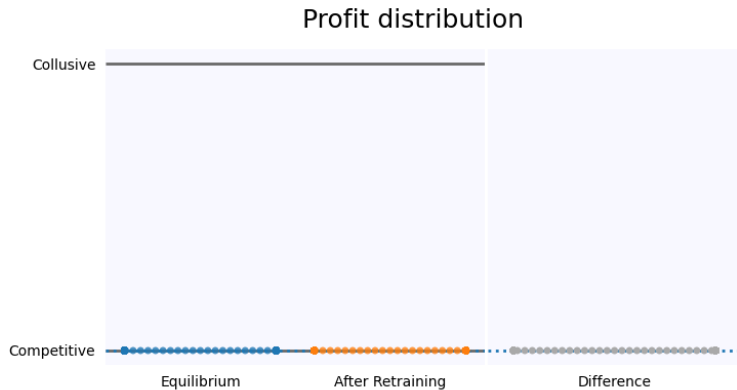


▶ prices

Outcome: consistent pattern of profitable deviations.

Re-training

3. Q-learning algorithm vs firm playing static NE (placebo)



▶ prices

Outcome: no deviation from competitive outcome.

Discussion

- Results seem to suggest that collusive strategies are not SP
 - ▶ Indeed, not designed to
- Re-training could constitute a model-free test to detect algorithmic collusion
- But it's infeasible

Can we achieve the same outcome without re-training?

Use observational data

Idea: re-train from past observational data.

- Take observational data
- For each state and action, estimate payoffs from k most recent observe data points

$$\hat{\pi}_i(s, a) = \frac{1}{k} \sum_{t=T-k}^T \pi_{i,t}(s, a) \quad \forall s, a \quad (1)$$

- For each state, action and future state, estimate state transitions from k most recent observe data points

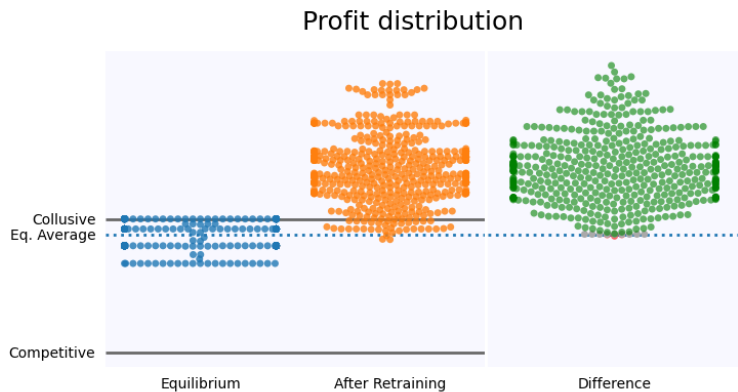
$$\hat{p}r_i(s'|s, a) = \frac{1}{k} \sum_{t=T-k}^T pr_i(s'|s, a) \quad \forall s, a, s' \quad (2)$$

- Given $\hat{\pi}_i(s, a)$ and $\hat{p}r_i(s'|s, a)$ compute $\hat{Q}_i(s, a)$

We cover the same 3 settings of the previous section.

Simulate re-training

1. Replication of Calvano et al. (2020).

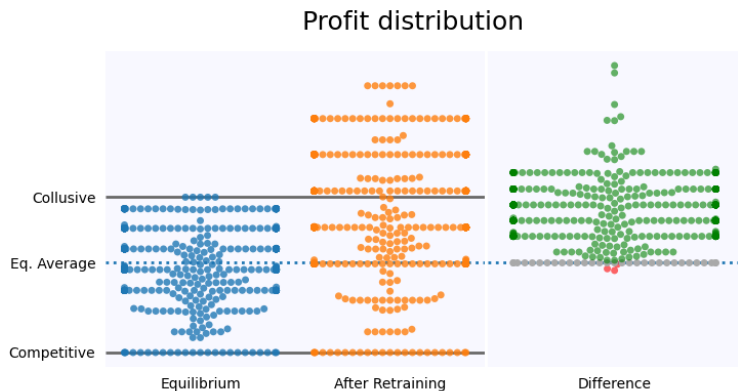


▶ compare with retraining

Outcome: consistent pattern of profitable deviations.

Re-training

2. Blind algorithms.

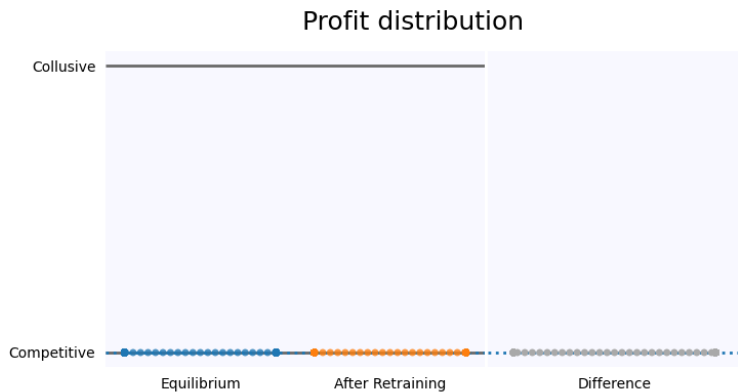


▶ compare with retraining

Outcome: consistent pattern of profitable deviations.

Re-training

3. Replication of Calvano et al. (2020) with $\delta = 0$.



▶ compare with retraining

Outcome: no sensible deviation from competitive outcome.

Conclusion

I have shown that

1. Even if one could inspect the algorithm strategy, collusive strategies that do not depend, directly or indirectly, on the rival behavior are possible
2. If algorithms were colluding, unilaterally re-training one of them leads to more competitive strategies
3. It is possible to implement this test unilaterally, using only historical data.

Conclusion

I have shown that

1. Even if one could inspect the algorithm strategy, collusive strategies that do not depend, directly or indirectly, on the rival behavior are possible
2. If algorithms were colluding, unilaterally re-training one of them leads to more competitive strategies
3. It is possible to implement this test unilaterally, using only historical data.

Thank you!

Bibliography

Bibliography I

- Asker, J., Fershtman, C., & Pakes, A. (2021). *Artificial intelligence and pricing: The impact of algorithm design* (Tech. Rep.). National Bureau of Economic Research.
- Assad, S., Clark, R., Ershov, D., & Xu, L. (2020). Algorithmic pricing and competition: Empirical evidence from the german retail gasoline market.
- Autorité de la concurrence & Bundeskartellamt. (2019). *Algorithms and competition*.
- Ballard, D. I., & Naik, A. S. (2017). Algorithms, artificial intelligence, and joint conduct. *CPI Antitrust Chronicle*, 29–35.
- Calvano, E., Calzolari, G., Denicolò, V., & Pastorello, S. (2019). Algorithmic pricing what implications for competition policy? *Review of industrial organization*, 55(1), 155–171.
- Calvano, E., Calzolari, G., Denicolo, V., & Pastorello, S. (2020). Artificial intelligence, algorithmic pricing, and collusion. *American Economic Review*, 110(10), 3267–97.
- Capobianco, A., & Gonzaga, P. (2017). Algorithms and competition: Friends or foes? *Competition Policy International*, 1, 2.
- Chen, L., Mislove, A., & Wilson, C. (2016). An empirical analysis of algorithmic pricing on amazon marketplace. In *Proceedings of the 25th international conference on world wide web* (pp. 1339–1349).
- Deng, A. (2017). When machines learn to collude: Lessons from a recent research study on artificial intelligence. Available at SSRN 3029662.

Bibliography II

- Ezrachi, A., & Stucke, M. E. (2017). Artificial intelligence & collusion: When computers inhibit competition. *U. Ill. L. Rev.*, 1775.
- Ezrachi, A., & Stucke, M. E. (2019). Sustainable and unchallenged algorithmic tacit collusion. *Nw. J. Tech. & Intell. Prop.*, 17, 217.
- Fershtman, C., & Pakes, A. (2012). Dynamic games with asymmetric information: A framework for empirical work. *The Quarterly Journal of Economics*, 127(4), 1611–1661.
- Gal, M. S. (2017). Algorithmic facilitated coordination: Market and legal solutions. *CPI Antitrust Chronicle*, 1–7.
- Harrington, J. E. (2018). Developing competition law for collusion by autonomous artificial agents. *Journal of Competition Law & Economics*, 14(3), 331–363.
- Klein, T. (2019). Autonomous algorithmic collusion: Q-learning under sequential pricing. *Amsterdam Law School Research Paper*(2018-15), 2018–05.
- Mehra, S. K. (2015). Antitrust and the robo-seller: Competition in the time of algorithms. *Minn. L. Rev.*, 100, 1323.
- OECD. (2017). *Algorithms and collusion: Competition policy in the digital age*.
- Okuliar, A., & Kamenir, E. (2017). Pricing algorithms: Conscious parallelism or conscious commitment? *Competition Policy International*, 1–4.

Bibliography III

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

UK Competition and Markets Authority. (2021). *Algorithms: How they can reduce competition and harm consumers*.

Appendix

Equilibrium prices before and after re-training

1. Replication of Calvano et al. (2020), conditional on collusion.

`figures/retraining_bothcoll_price.png`

Equilibrium prices before and after re-training

2. Blind algorithms, high competition.

figures/retraining_selfcoll_price.png

Equilibrium prices before and after re-training

3. Blind algorithms, low competition.

`figures/retraining_selfcomp_price.png`

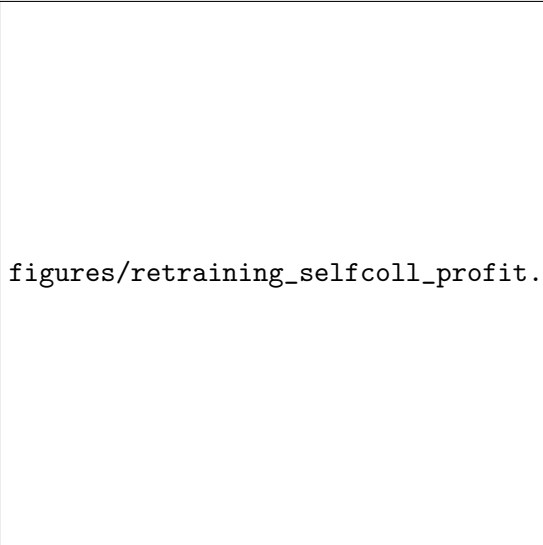
Compare re-training and simulated retraining

1. Replication of Calvano et al. (2020), conditional on collusion.

figures/retraining_bothcoll_profit.png

Compare re-training and simulated retraining

2. Blind algorithms, high competition.



figures/retraining_selfcoll_profit.png

Compare re-training and simulated retraining

3. Blind algorithms, low competition.

figures/retraining_selfcomp_profit.png