Regulating Algorithmic Collusion

Vikram Maheshri, Ankit Patel & Ryan Pyle

March 6, 2025

Motivation

- A fundamental economic question: How should I set my price?
- As markets become more complex, perhaps we should turn to algorithms to help with this.
- Given recent advances in computing/AI, perhaps we should let our algorithms learn to do better.

Motivation

- We have insights into what happens with human sellers compete.
  - Benefits to competition
  - Sometimes these benefits are threatened. How can regulators preseve these benefits?

  What happens when algorithms compete?
  - Do we still obtain benefits to competition?
  - If these benefits are threatened, is regulation feasible? Effective?

Background on Algorithmic Competition

- Calvano et. al. (2020) (CCDP) pitted two very simple reinforcement learning algorithms in repeated Nash-Bertrand competition.
- The algorithms don't know any economics. They see a history of prices and payoffs and spit out a price.
- After enough play, the algorithms learned to collude on average! 40-50% excess profits.
- Skeptics (e.g., Kuhn and Tadelis (2018)): It takes too long, so it's not an issue in practice.
- Assad et. al. (2023) show that the use of algorithmic pricing in German retail gas markets led to 15-20% increaeses in price.
- US DOJ Antitrust Division: We're hiring a bunch of computer scientists and we're bringing some high profile cases (e.g., RealPage).
  - Strategy is still the same... we're going after communication.

In this paper, we show:

1. Basic modifications to the CCDP algo lead to several orders of magnitude improvement in learning speed and greater collusion at steady state.
2. These improvements follow from a more sophisticated learning process
   - Collusion doesn't just emerges on average when algorithms compete
   - A single agent learns to collude in a swift and disciplined way
3. These algorithms can be audited, which yields insights into how collusion is learned.
4. We use these insights to propose a fundamentally new approach to regulation based on latent influence ("intent") as opposed to observed action (or communication).
   - Influence-based regulations outperform action based regulations and are more robust to changes in the market environment.

AI, Algorithmic Pricing and Collusion (CCDP 2020)

- Canonical collusion model – an infinitely repeated pricing game, all firms act simultaneously, actions conditioned on price/profit histories.
- CCDP assume bounded memory – we can relax this.
- n differentiated products and an outside good. Each product is supplied by a different firm. Basic logit demand with product quality indices, an index of aggregate demand, and an index of horizontal differentiation.
- Key object of interest: "profit gain", i.e., excess profits relative to Nash-Bertrand equilibrium.

AI, Algorithmic Pricing and Collusion (CCDP 2020)

- Each firm is represented by a Q-learning algorithm.
- Not obvious *ex ante* how the algos will behave.
- Convergence not guaranteed.
- Let them play!
- Do it many times and average results.

Q-learning

- The Q-function is like a precursor to the Bellman Equation. Instead of

$$V(s) = \max_a E[\pi|s,a] + \delta E[V(s'|s,a]$$

  we have

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

- If the agent knew the Q-matrix ($|S| \times |A|$ matrix of values of the Q-function) then it's easy to determine the optimal decision.
- Q-learning algorithms estimate the Q-matrix iteratively. With some probability, *exploit* and take the best option given current beliefs. Otherwise, *experiment* and try something new.

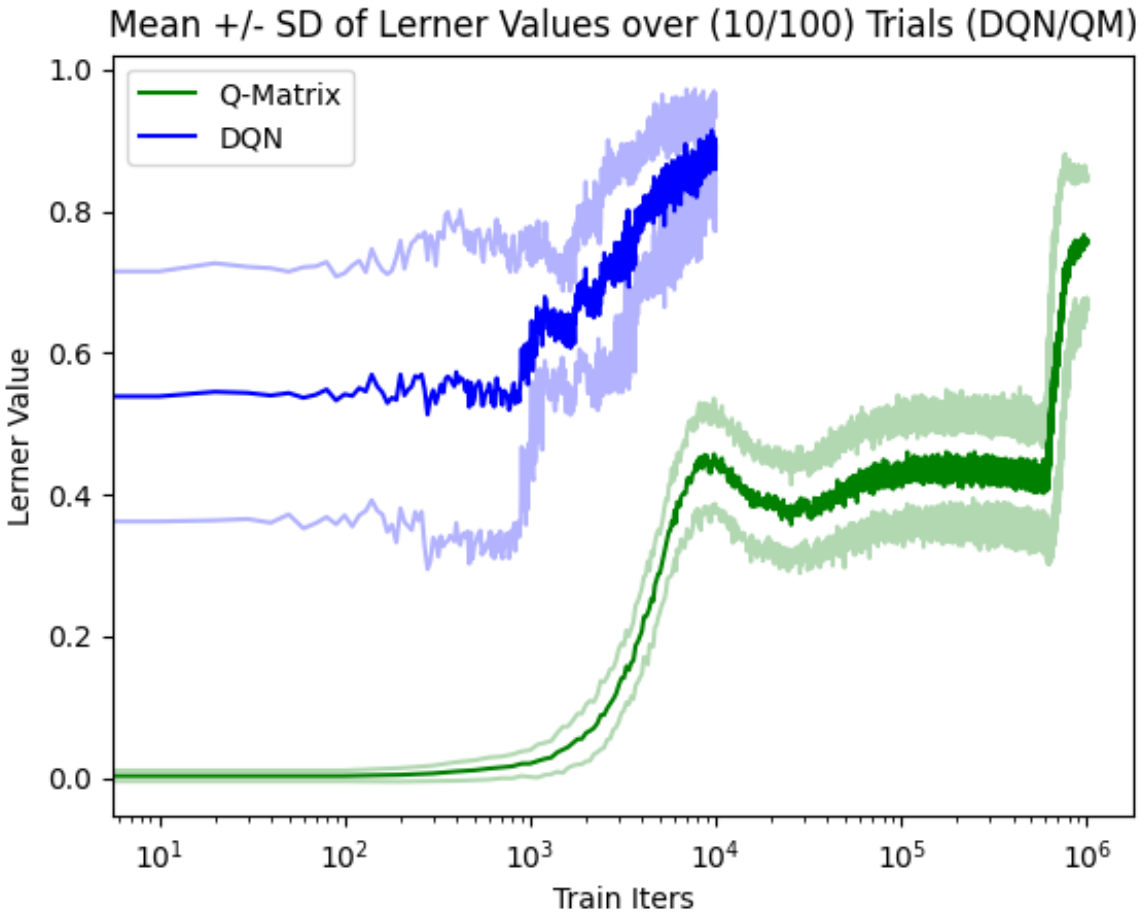
Set Up and Results

- Simple set up: 2 agents, 15 possible states (prices) per agent
- Agents see previous price of both agents, choose next price. (Q-matrix is $15^2 \times 15$)
- Run it a bunch of times in parallel, see if/when it converges on average.
- Takes a long time - each incoming data point can only update one cell of the Q-matrix.

Our Approach - DQN

- Deep Q Network: Approximate the Q-matrix with a neural network
- Allows you to learn non-locally
  - CCDP algo only updates one entry of the Q-matrix per play
  - We adjust the weights of the neural network after every play, so many entries can in principle be updated
- Somewhat obvious: yields large increases in learning speed
- Less obvious: converges to a different, more collusive equilibrium
- Even less obvious: collusion learned at the agent level, does not emerge as an aggregate phenomenon

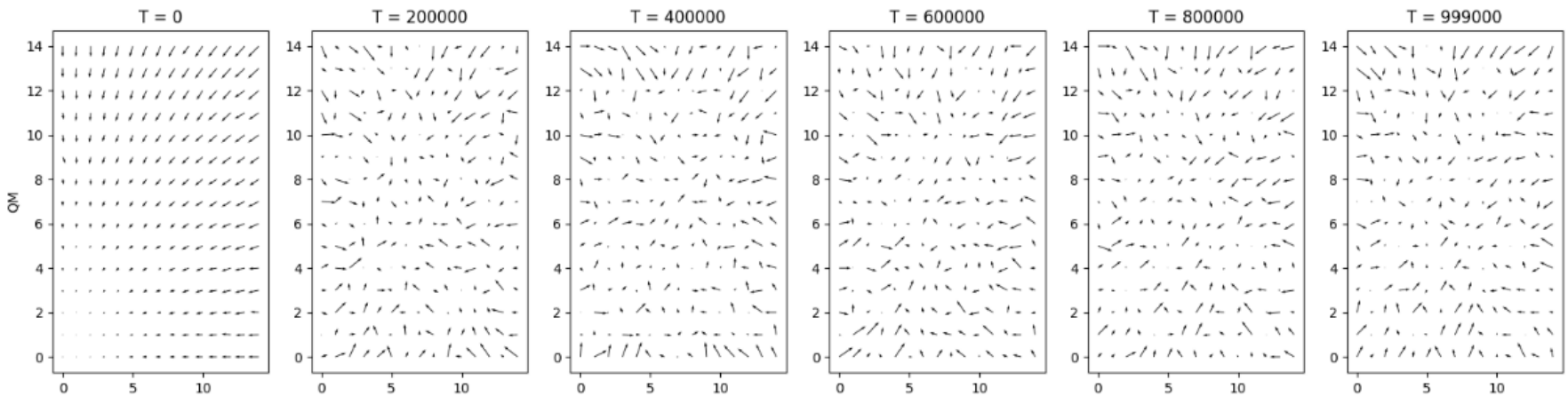
DQN v. QM



Much faster, more collusive!

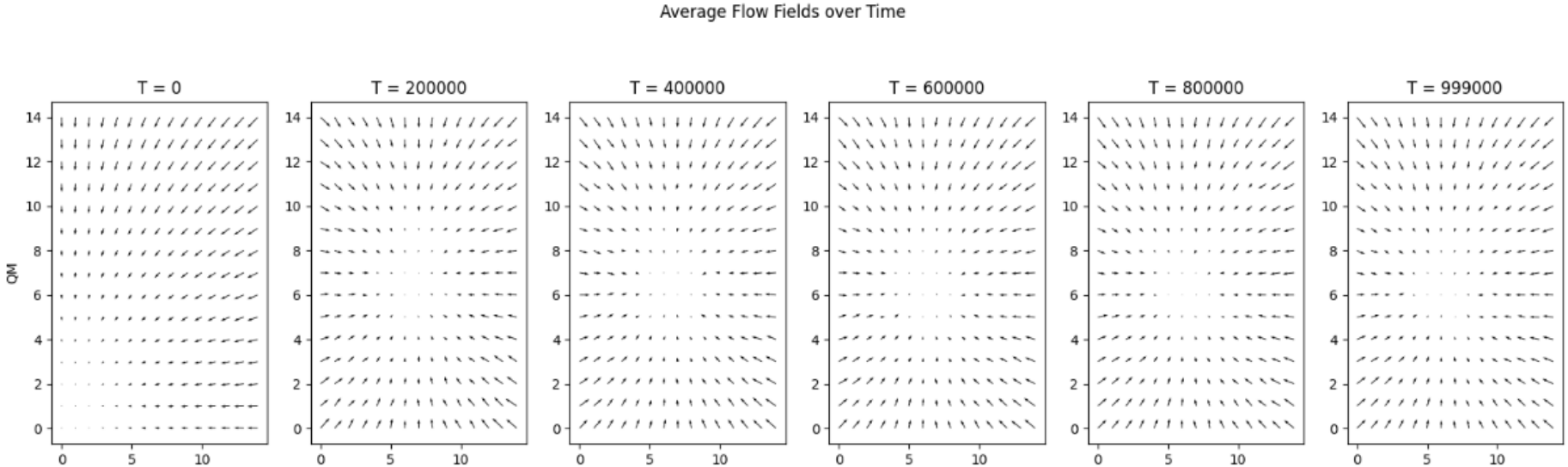
Learning in the QM

Learning is haphazard.



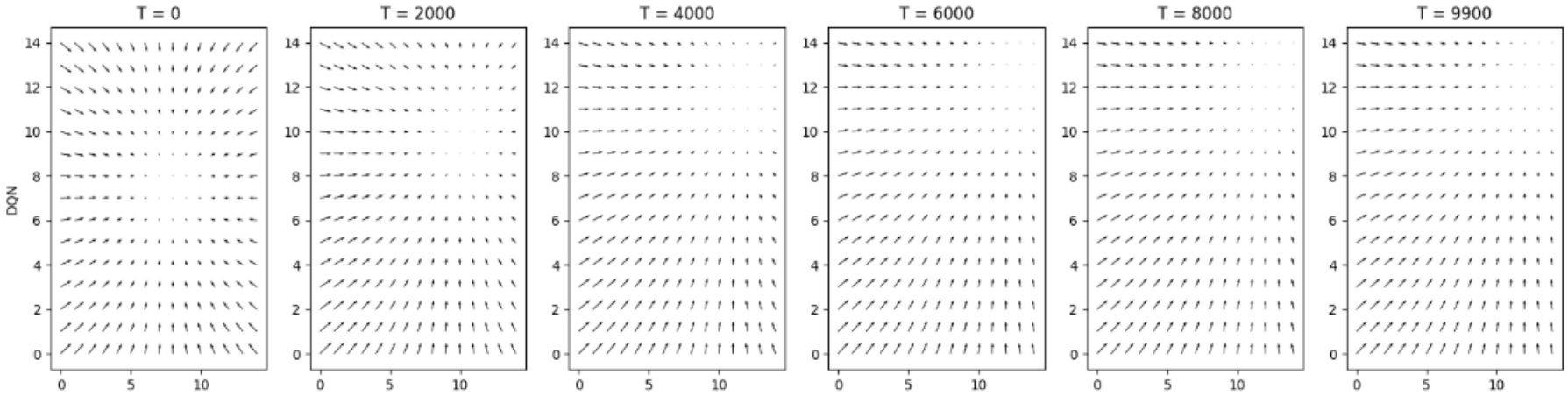
Learning in the QM

Collusion only emerges in aggregate (100 trial avg.)



Learning in the DQN

Learning is well behaved at the individual level.



Robustness

- This is a very stylized set up. Let's consider a variety of real-world modifications:
  - More than 2 firms
  - Aggregate demand shocks
  - Uncertain product quality
  - More horizontal differentiation
  - Bigger state space
  - Longer memory
- DQN advantage over QM widens in all cases.
- In some cases, complications can even improve the DQN's *absolute* performance!

Auditing Collusion

- We perform white-box audits of the QM and DQN algos.
- Open up the black box to learn:
  - How does the algo learn?
  - How does the algo incorporate new information?

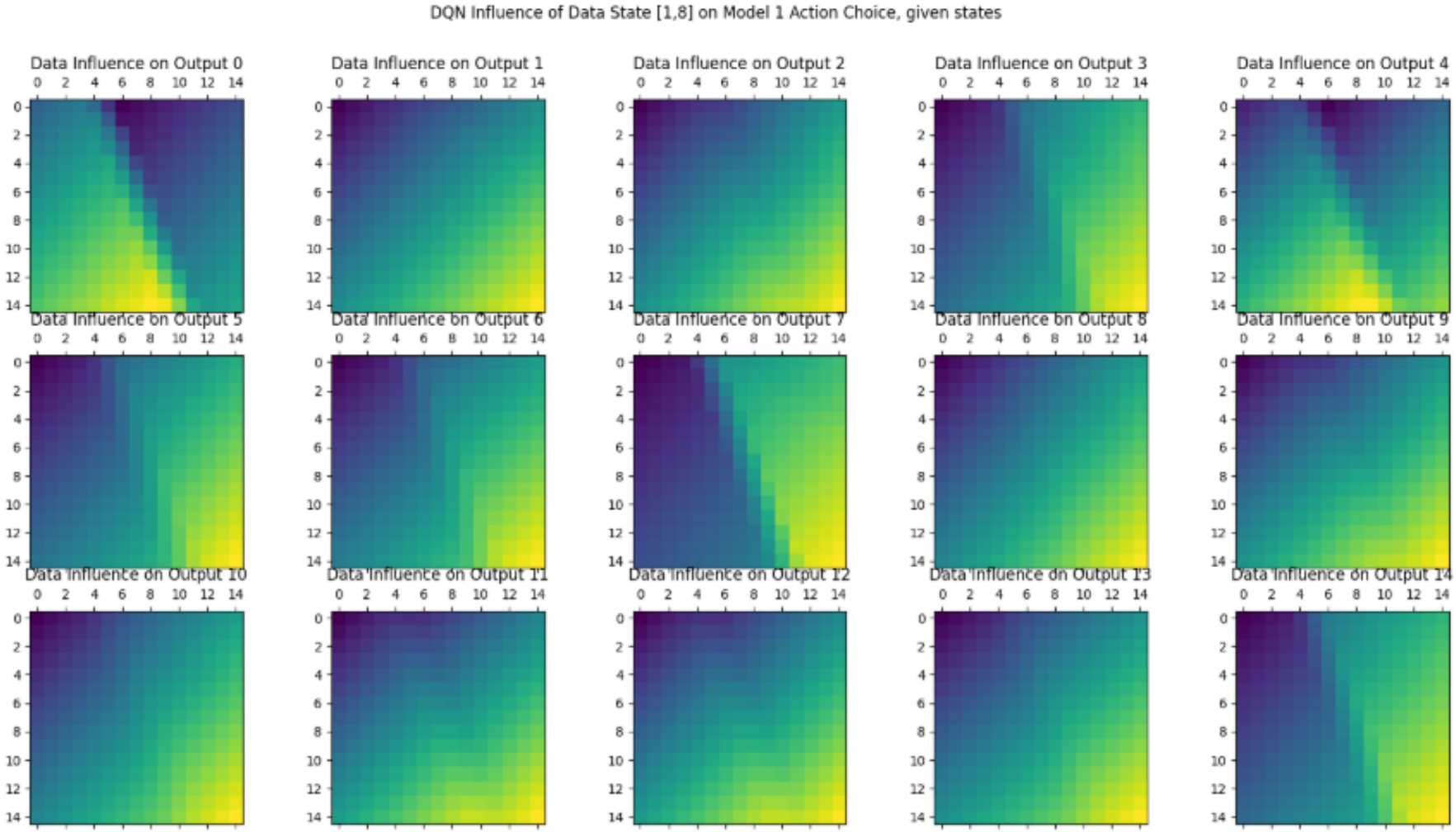
Auditing Collusion: 10,000 Foot View

- The DQN is essentially takes in data, passes it through a series of layers, and spits out some function of the data.
- Each layer does a series of simple non-linear (ReLu) transformation of the data governed by a series of weights.
- WIth each additional observation of training data, the weights are tuned.
- The audit allows us to see which instances (i.e., observations in the training data) most influence this tuning process.
- We can also compare how the different algos incorporate new info vs. rely on learned knowledge.

Main Audit Results

- The QM learns in a halphazard/jumpy way.
  - New data is not systematically incorporated in the QM.
  - Collusion is largely influenced by the state in the training data.
- The DQN learns in a smooth way.
  - New data is systematically incorporated in the DQN.
  - Collusion is largely influenced by (recent) action.

Visualizing Some Audit Results



Can we do anything about this?

How does the standard antitrust toolbox to regulate algorithmic collusion?

The process that produces higher prices

	COMMUNICATIONS	COLLUSIVE PRICING RULES	HIGHER PRICES
Humans	Present, discoverable	Latent, not discoverable	Observable, difficult to evaluate
Algorithms	Not present	Latent, discoverable	Observable, difficult to evaluate

Regulation Within an Algorithm

- Regulating human decisionmakers is necessary after-the-fact.
- Regulating an algorithm could be done within the decision-making process.
- In this context, a regulation can be a constraint on how the algorithm learns.
- We consider two regulations:
  1. Behavioral - constrain learning from bad behavior
  2. Influential - constrain learning from bad ”intentions”
- Constraints are implemented as a few extra lines of code.

Behavioral Algorithmic Regulation

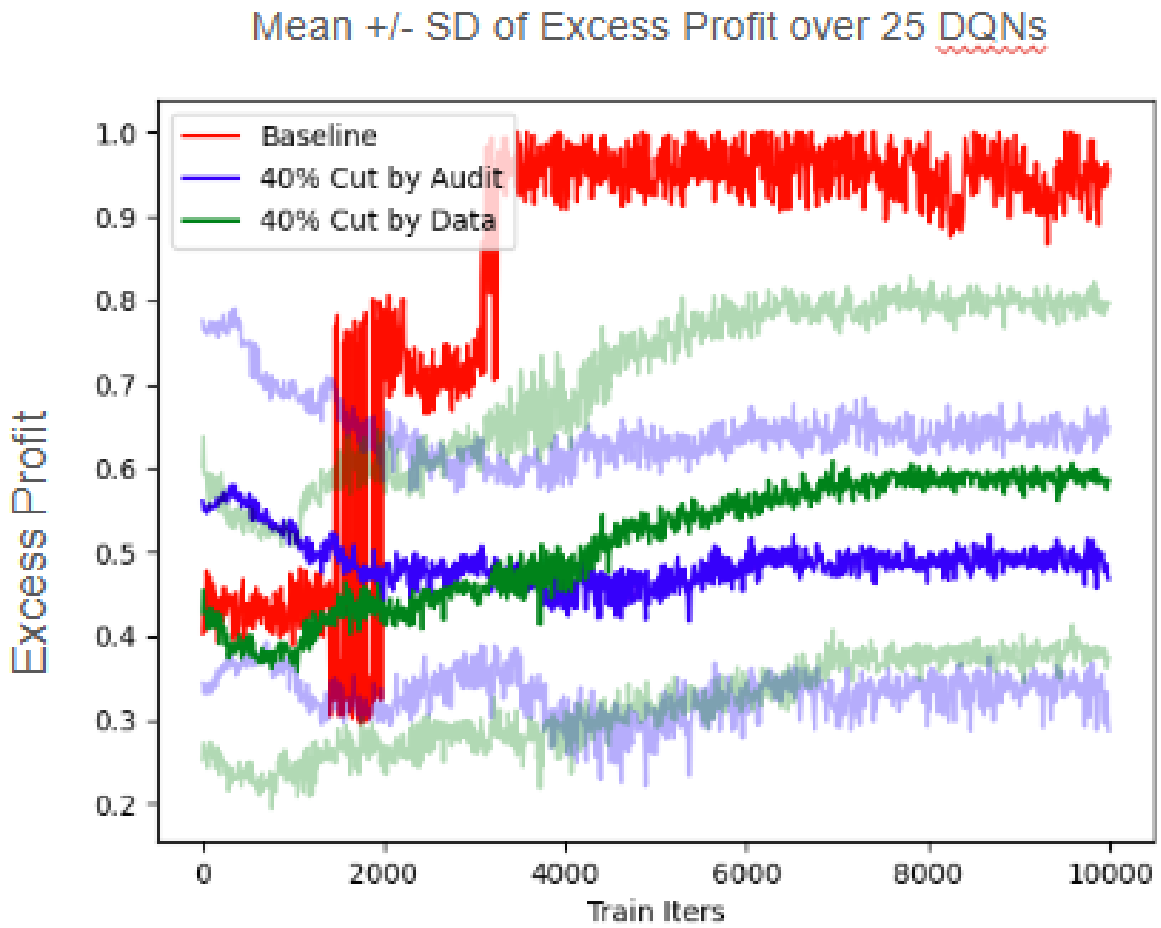
- Allow algorithms to play as before.
- If some function of the two algos' excess profits in a particular stage of play exceeds some threshold, then do not allow the algo to learn from this instance.
- How do you implement this if you can't determine excess profits?
- What function of excess profits is the right one?

Influential Algorithmic Regulation

- Allow algorithms to play as before.
- If in some instance causes the Q-matrix of an algo to shift in a sufficiently collusive manner, then do not allow the algo to learn from this instance.
  1. Compute the delta-Q flow field
  2. Sum the deltas weighted by the implied change in profits
  3. Restrict learning if that's too big
- Can be implemented without knowing excess profits
- Can be differentially applied to the different competing algos

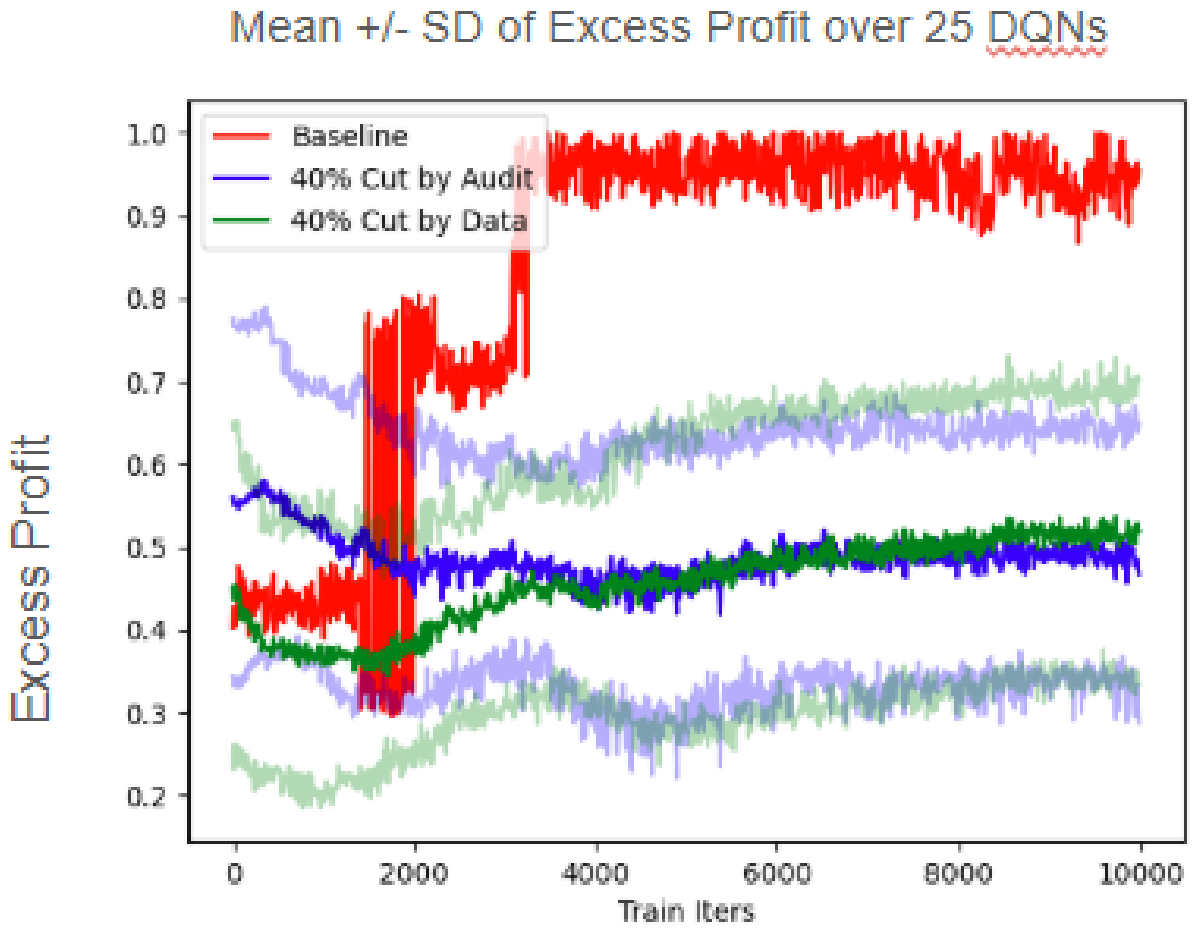
Behavioral v. Influential Regulation 1

- In both cases, restrict 40% of training observations.
- Behavioral restriction: threshold determined by total excess profits



Behavioral v. Influential Regulation 2

- In both cases, restrict 40% of training observations.
- Behavioral restriction: threshold determined by minimum excess profits



How to implement these regulations in the real world?

One proposal:

1. Require any pricing algorithms to be registered with the regulator before it goes live.
   - Precedent: HFT algos must be pre-registered with SEC.
2. Set up a behavioral screen in each market. If observed prices grow too fast, then an alarm is triggered. Cheap.
3. If an alarm is triggered, then insert regulatory code into the algos.

Conclusion

- Advances in computing suggest that we should take algorithmic collusion seriously.
- We need a new regulatory framework to deal with algorithmic collusion – computers don't talk like humans do.
- Auditing AIs opens up a fundamentally different approach to regulation.
- Will algos learn to defeat our simple proposed regulations?
  - Maybe... Probably... Almost certainly :(
- The next step would be to develop an algorithmic regulator.
- Like all enforcement problems, it's a cat and mouse game.