Can Reinforcement Learning in Markets and Auctions lead to Algorithmic Collusion?

May 15, 2023

1 Introduction

1.1 Overview

Humans are increasingly handing off decision-making to algorithms. These decisions can be simple like handling files or storing data or very complex like processing text and managing a portfolio. Reinforcement Learning (RL) is a branch of AI that deals with autonomous decision-making under uncertainty. The key idea is to learn the optimal response through an exploration of the environment and accurately mapping the value of actions taken. Reinforcement learning, armed with deep neural networks, has superhuman performance in complex single and multi-player games like Chess, Go, Atari Games and Starcraft. Beyond video games, there are many modern applications in economic decision-making such as pricing, bidding, advertising, and trading. The rise of algorithms like reinforcement learning in markets leads us to question if current structure and design of markets is adequete in ensuring competition. Algorithmic collusion is when algorithms learn to collude without any human interference and communication. For instance, this could mean suppressing bids or rotating bids in ad-auctions or double call auctions. Another example could be the systemaic rise in prices by pricing-algorithms on ecommerce websites.

The fields of computer science, operations research, and even economics are showing great enthusiasm in bringing reinforcement learning to markets. Consider the following highly cited papers in the last five years papers. Deng et al (2016) use a Deep Recurrent Neural Network (RNN) to parse financial data on stock and futures and use reinforcement learning to learn optimal trading strategies. Lu et al (2018) use an reinforcement learning algorithm to handle demand response to energy demand-supply mismatches. Q-learning is used to solve for optimal dynamic pricing in a hierarchical electricity market. Cai et al (2017) build a Deep RL model for real-time bidding in online ad auctions - to handle both valuations of ads and strategically bid against opponents. Their model is successful in a live A/B test. Zou et al (2019) incorporate RL into recommender systems. They model user behavior in an LSTM and use RL to optimize user engagement through its recommendations. Mishra et al (2019)

imbed microeconomic choice theory into a multi-armed bandit to minimize the cost of price experimentation. They demonstrate the success of this method in a field experiment.

1.2 Empirical Evidence

There is large evidence for the pervasive adoption of algorithms in markets. Chen et al (2016) study 1,641 best-seller products on Amazon and detect that about 543 had adopted algorithmic pricing. A 2017 EU survey found that "Two thirds of them [ecommerce firms] use automatic software programmes that adjust their own prices based on the observed prices of competitors." A 2023 eMarketer report shows that algorithms are dominating bidding in display and sponsered search auctions accross the globe. Brogaard et al 2014 find that High frequency Trading (HFT) algorithms have dominated trading in stock markets. Tao et al 2021 review the rise of robo-advisors to manage portfolios and reduce expenses. Revenue management is commonly used in airlines, hospitality, rental and advertising for dynamic pricing and availability of a finite inventory that sells in a particular season. It should be highlighted that most of these algorithms are not as sophisticated and autonomous as reinforcement learning - most just use simple rules with feedback from their competitiors and customer demand.

There is also some preliminary evidence on the possibility of algorithmic collusion. Assad at al (2020) study Germany's Gasoline market and show that adoption of pricing algorithms increases average margins by 9% in competitive markets and 28% in duopolies. Brown and Mackay (2021) find that firms with better algorithms update their prices faster and keep them higher. Cavallo et al (2019) show that products on Wallmart that are also on display at Amazon remain on the shelf 20% lesser time. Furthermore, there is also much simulational evidence that algorithmic collusion is easily attainable in a variety of pricing and bidding games.

1.3 Regulatory Issues

With the rise of pricing algorithms come court cases related to them. The Department of Justice (DOJ) in the US has been busy. In 1994, DOJ prosecuted six airlines that used a common online booking system. This is the most famous case of collusion through pricing algorithms. Airlines were able to carry out private dialogues via the forum. In 2017, the DOJ began an investigation into RealPage which designs rental algorithms. In 2016, the DOJ charged two competitors for designing pricing algorithms on Amazon Marketplace that would undercut the rest of the market but not compete any further.

As algorithms get smarter, this situation is even more concerning legally. When self-learning algorithms learn to charge higher prices or suppress bids purely by themselves it falls under "tacit collusion" and that is generally not prosecuratable. Tacit collusion i.e price coordination without communication

is not covered under US Sherman Act. The US law requires evidence of "actionable agreement" over mere interdependent behavior. Firms are free to build algorithms that incorporate current and past information about other firms' prices and price algorithms. European Law (Article 101 TFEU) outlaws three types of collusion: agreements, decisions, and concerted practices. Again, this does not include Tacit collusion.

The goal of this paper is to highlight the ability of reinforcement learning algorithms in autonomously discovering collusion in pricing and bidding games. In particular, we will examine the key mechanism that permits this to happen. We first exposit the basic idea behind reinforcement learning and then review previous studies. Then we will demonstrate through a series of simulations how collusion can be supported. We will conclude with a discussion of what can be explored next and what are other issues need to be highlighted.

2 Reinforcement Learning

2.1 Thought Experiment

The idea behind reinforcement learning can be explained simple by a thought experiment. Consider a horror movie where the protagonist is repeatedly dropped off in a labrinth/maze. However the tiles of this labrinth are uniquely numbered - allowing the protagonist to keep track of where they are dropped and where they move to. The protagonist must explore this maze and find the exit - where she can return to her family for a week - before this process restarts. There are some booby traps laid along the way and if the protagonist dies she is resseructed randomly anywhere in the maze. There is also some food kept in some locations. There is only one exit and the maze does not change its base structure. The protagonist has a good memory and carries a small notebook which retains all notes made in the previous life. What happens in this horror movie? The first few times the protagonist gets killed. But she begins to make progress - by noting down where the traps are laid and learns to avoid them. She also learns to find the food quickly. Finally she discovers the exit! This opens a new saga - the protagonist now yearns to find the exit using all the information previously gleaned. The protagonist must avoid traps, find food and most importantly, find the exit.

While this is a rather dismal movie - but it highlights what is going on. The agent begins with a blank slate - they know nothing about the environment. The environment is recorded through the tiles of the labrinth - they register "where" the agent is at. The actions that the agent can take are to move - left, right, forwards or backwards. Of course, not all actions are possible at all locations. The rewards in certain locations can be positive (food) and negative (booby traps) and overwhelmingly positive (exit). The environment is deterministic and fixed because the rewards locations do not change. The agent must take risks and explore to discover rewards - the booby traps and food is not known in advance. The "policy" that the agent learns is how to move given where they

are located. The agent must also record their positive and negative experiences - this allows them to avoid the negative and obtain the positive. The moment the agent discovers the exit - all future attempts are now redirected towards finding it. Initially the agent suffers a lot and spend a lot of time stuck in the maze - but over time the situation changes. The agent is able to efficiently find the exit repeatedly - they learn the optimal response of where to move given where they are.

2.2 Mathematical Framework

More formally the problem of the agent is described by a Markov Decision Processes (MDP). A MDP is (S, P, R, γ, A). S is the set of possible states (e.g. coordinates of the maze). A Markov Process has the characteristic A is a set of actions (e.g. left, right, up, down). R is reward matrix that measures $R(s, a, s')$ or the reward obtained at state s, taking action a and moving to s'. P is transition matrix that measures $P(s'|s, a)$ or the probability of moving to state s' when taking action a at state s. And γ is the discount factor. Agents follow policies to move through the environment. A policy π is a strategy of choosing actions given states such that $\pi_{a,s} = Prob(A_t = a|S_t = s)$. Policies can be random as well as deterministic by having a fixed mapping between states and actions.

2.2.1 Bellman Equations

Every policy determines a series of possible rewards. Some policies are better than others and are able to extract more reward on average than other policies. The goal of reinforcement learning is to learn the optimal or "best" policy. But for that we need a way to value each policy. Let $\pi(s_t)$ be the action taken at time t according to policy π. Then we the discounted reward from starting out at state s_0 is going to be $\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t))$, the return from following policy π onwards from state s_0 in one history. Better policies yeild better returns from initial state. However the environment is stochastic and each history would be different from another, so we must compare expected return instead. Thus for each policy π we can "value" states :

$$V^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) \mid s_0 = s; \pi\right]$$

where movement through the states are governed by the transition function $s_{t+1} \sim p(\cdot \mid s_{t-1}, a_t = \pi(s_t))$. The $V^{\pi}(s_0)$ function tells you the expected return from following π at s_0. This formulation is useful when the reward function and transition probabilities are known - for it is a simple matter to start off at state s_0 and then just simulate histories and compute the average return over all of them to evaluate any policy. However, this formulation is not so useful when the reward and transition function are unknown but we only have access to the reward and transition between states one at a time. This would be accurate

in a "online" setup where we actually drop the agent in a world and let them experience rewards and see transitions one at at time. In this scenario, which is emblematic of many real-world situations, we pull out the first reward (which can be experienced) and leave the rest to be approximated.

$$Q^{\pi}(s,a) = r(s,a) + E\left[\sum_{t=1}^{\infty} \gamma^t r(s_t, \pi(s_t)) \mid \pi\right]$$

$Q^{\pi}(s_0, a)$ allows us to study the impact of choosing different actions a at s_0 even while continuing to following π after the first step. The Q table is necessarily larger than the V table because it also contains an evaluation of each action given each state. However we should note that $Q^{\pi}(s_0, \pi(s_0)) = V^{\pi}(s_0)$. Both Q and V allow us to compare and evaluate policies but the Q function is easier to approximate when information is revealed one step at a time.

There can be an infinite number of possible policies but only one can be the best (as long as the maximum expected return is not infinite). When we look at the V and Q of the optimal policy we see that they follow a special recursive representation - these are the Bellman Equations.

$$V(s) = \max_a r(s,a) + \gamma E[V(s')]$$

$$Q(s,a) = r(s,a) + \gamma E[\max_{a'} Q(s', a')]$$

such that $s' \sim p(\cdot \mid s, a)$

In the first case, the best case valuation of state s is going to be choosing the best action a^* that maximizes the sum of reward in current period with the discounted expected value of the best case valuation of the next state. In the second case, the best case valuation of taking action a in state s is the reward from (s, a) and the discounted expected maximum best case valuation from the next state. These functions can also be approximated, under mild conditions, in an iterated way where a guess is used to compute the right hand side of the equation and the guess is updated from the left hand side - and on till the guess convergences.

2.2.2 Q-Learning

One of the most popular algorithms to approximate the Q function is the Q-learning algorithm which can be operationalised in an "online" setup.

- Guess $Q_0(s, a)$
- at t, do:
  - observe s_t
  - take action a_t from exploratory strategy
  - collect reward r_t
  - observe transition s_{t+1}

– update Q at point (s_t, a_t):

$$Q_{t+1}(s_t, a_t) = (1 - \alpha)Q_t(s_t, a_t) + \alpha(r_t + \gamma max_{a'} Q_t(s_{t+1}, a'))$$

2.2.3 Exploration and Hyperparameters

Exploratory strategies are methods to take actions given states. they need not be the same as what the Q table currently recommends (that is the greedy policy). We do this becasue initially the Q table is expected to be far off the mark and only gradually becoming accurate. The following are some exploratory Strategies, that pick a_t given s_t:

- Random: $P(a_t|s_t) = 1/|A|$
- Greedy: $a_t = argmax_a Q_t(s_t, a)$
- ϵ-Greedy: Random with $1 - \epsilon$ and greedy with ϵ
- Boltzmann Exploration:

$$P(a_t|s_t) = \frac{e^{Q_t(s_t,a_t)/\beta}}{\sum_{a'} e^{Q_t(s_t,a')/\beta}}$$

Q-learning also comes with some key hyperparameters that govern the overall exploration and exploitation process. The first is the learning rate α which is the weight given to current experiences over past. If α is high then current rewards make large adjustments to the Q table and learning is fast (but inaccurate). When α is low then the Q table current valuation is repected for a long time and current rewards make a small impact on the Q table. This would imply that learning is slower (but accurate). To ensure good performance, the learning rate must be low enough for sufficient exploration to happen and high enough for discovery of high rewards be able to influence the Q table. In fully deterministic environments a learning rate of 1 is optimal. In stochastic environments, usually a value is chosen between 0.1 and 0.00001.

The second hyperparameter is the initial valuation Q_0. This matters because it usually influences initial exploration. Usually this should not matter much but when the table is very large and exploration may not visit each state-action pair many times we would need to put an intelligent initial valuation to ensure we are exploring well in the beginning. The third set of parameters govern the probability of random exploration ϵ. This includes the starting and end points for ϵ as well as its decay rate. Decay too fast and you do not explore enough outside the dictates of the Q table and decay too slow and learning takes too long.

Another set of parameters govern the start, end and decay rate of the intensity of explorative-exploitation β. In Boltzmann exploration we are using the Q table to guide exploration - something that is important when the state and action space is large and high dimensional. When β is high we behave almost randomly but when β is low the Q table filters out all but the top candidate

actions. And finally when β is very low, only one action is chosen i.e. greedy exploration. Exploration is essential because in the beginning the initial valuations are wrong and we need to behave randomly to explore the world. Once our valuations improve we can cease to behave randomly and be guided more and more by our experiences which are imprinted in the Q table.

2.2.4 Demos

There are two interesting visual demos that can be used to understand Q-learning. The first is a mouse travelling through a maze. It can be found here. The second is a much more complex simulation between agents playing hide-and-seek. It can be found here.

2.3 Salient Points

There are also a couple of salient points about Q-learning. Firstly, it is model-free - in the sense that we do not need to know thetansition matrix P and the reward R matrix. What we do need to have is the ability to play the game in an online fashion and collect rewards and transitions by taking actions. Secondly, Q-learning is an off-policy learning algorithm. This means that it evaluate policies from off-policy routes i.e. not following the policy that it is actually evaluating. For instance, we could just act completely randomly in the exploration step and this would still help us populate the Q-learning table in the update step (even if we never use this udpate step). This would be inefficient - akin to bumbling about forever in the labrinth - but doable. Thirdly, Q-learning is incremental and slow. It allows for large initial Bellman errors i.e very poor guesses in the beginning - often taking wild and large detours and suffering heavy losses - and taking a long time to finally correct these mistakes. It is slow because it updates one state-action value at a time, making no attempt to infer the valuation error at any other state-action pair.

Fourthly, Q-learning comes with an important guarentee. Watkins and Dayan (1992) show that with sufficient exploration and a stationary environment we will get to the optimal Q with probability 1. Thus despite being slow, it gets the job done. Fifthly, Q-learning is very practical and overcomes the main hurdles that smarter strategies of solving this problem cannot. Dynamic programming is often impossible because modellers do not know the reward and transition function. On-policy learning that updates the same policy that it is exploring are very sensitive to the initial conditions and can get stuck in a policy that is suboptimal in a long term sense. Other policy learning strategies that update the Q-table at all places may wrongly do so and lead to a misleading valuation. Sixthly, the guarentee that Q-learning will find the optimal policy only works in stationary environments. This can include highly stochastic and evolving environments - but ones which have some form of regularity i.e. patterns may be wild but will ultimately repeat. Once the environement is non-stationary i.e. that initial conditions will ultimately govern the long term behaviour, then Q-learning is unable to find the optimal policy as it is chasing a

moving target. This guarentee also is limited to the single-agent case and does not mean that multiple agents interacting and influencing each others' transitions will be able to find the optimal policy. To be sure, to not be able to find the optimal policy does not mean that you cannot find a good or good enough policy - one that is adequete from a problem solving standpoint.

2.4 Enhancements

While the base algorithm is able to solve simple problems with less randomness and low dimensional action and state spaces. The first issue arises when state-action space becomes very large. In such a situation, Q-learning becomes too slow to be practically useful. The second issue is because the Q-learning algorithm must compute the expected maximum of the Q-table on the RHS, it has an overestimation bias i.e. in noisy environments it overvalues certain states. This overestimation is ultimately corrected but can slow the overall learning process. The third issue is that Q-learning is designed for single-agent problems and not for multi-agent problems. In multi-agent problems we may have both adversarial games as well as cooperative games.

2.4.1 Deep Q-Learning

The first issue of flexible representation is overcome by the use of deep neural networks to represent a many-to-many mapping between the states (which can be multi-dimensional and continous) and the Q-values of all possible actions (which is generally discrete and low-dimensional). Thus if $s \in S$ is the state space we use a neural network to approximate $Q(S;\theta)$ as a $|A|$ dimensional vector representing the Q-values of actions taking in that state. The training of neural networks requires larger samples than just the one instance used to update in original Q-learning. Thus learning takes place by sampling from history of experiences - and batches of state-action-reward-transition data. The Deep Q-learning Network algorithm works as follows:

- Initialize: $Q(s, a; \theta)$ with a neural network.
- At t
  - Choose a_t from exploratory strategy
  - Observe s_{t+1}, r_t
  - Draw a random sample $(\bar{s}, \bar{s}', \bar{r}, \bar{b})$ from history
  - Construct Bellman Error: $e = Q(\bar{s}, \bar{a}) - (\bar{r} + \gamma E[max_{a'} Q(\bar{s}', a')])$.
  - Loss: $L(\theta) = e'e$.
  - Backpropagation: Compute $L'(\theta)$.
  - Update Parameters: $\theta = \theta - \delta L'(\theta)$.
  - End when loss does not decrease anymore

2.4.2 Double Q-Learning

The second issue of overestimation is overcome by a process called Double Q-Learning. Here we split the Q table into a copy and using one Q-table to calculate the expected maximum in the RHS and another to update the value of state-action pairs. Both tables refer to each other and are updated over time. But this cross-referencing replaces the overestimation bias with an underestimation bias and keeps the networks more conservative.

- Guess $Q_0^A(s,a), Q_0^B(s,a)$
- at t, do:
  - observe s_t
  - take action a_t from exploratory strategy
  - collect reward r_t
  - observe transition s_{t+1}
  - update Q^A, Q^B at point (s_t, a_t):

$$Q_{t+1}^A(s_t,a_t) = (1-\alpha)Q_t^A(s_t,a_t) + \alpha(r_t + \gamma max_{a'} Q_t^B(s_{t+1},a'))$$

$$Q_{t+1}^B(s_t,a_t) = (1-\alpha)Q_t^B(s_t,a_t) + \alpha(r_t + \gamma max_{a'} Q_t^A(s_{t+1},a'))$$

2.4.3 Game Theoretic Q-Learning

Multi-agent Q-learning can be augmented by incorporating concepts from game theory and modelling opponents. And while it is possible to just use independent Q-tables $Q_i(s,a_i)$ to solve multi-agent games, we are likely to see a lot of variance in the rewards per action taken if we do not factor in the actions of others. This is done by using Q-tables that include joint actions $Q_i(s,a_i,a_{-i})$. This effectively means that each agent is tracking the valuation of others. Each agent i at each step computes the Nash Equilibrium using its own and its belief about others' Q-tables. This may be a mixed strategy Nash. After this step, each agent i knows her and others' expected return from the Nash equilibrium. This value is used to update the Q-tables in agent i's memory. In an N player game, agent i does:

- Guess $Q_i(s,a_i,a_{-i}), Q_{-i}(s,a_i,a_{-i})$
- at t, do:
  - observe s_t
  - take action $a_{i,t}$ from exploratory strategy
  - observe other strategies $a_{-i,t}$
  - observe rewards $r_{i,t}$
  - observe transition s_{t+1}

- use linear program to solve Nash Eqbm:
  - Stage game payoffs $u_j(a_i, a_{-i}) = Q(s_t, a_i, a_{-i})$
  - Mixed Strategy Nash: $(\pi_1, \pi_2, ...\pi_N)$
  - Value of MSNE to j: $\Gamma_j(\pi_1, \pi_2, ...\pi_N) = \sum_i u_j(a_i, a_{-i}) Q(s_t, a_i, a_{-i}) \Pi_{k=0}^{N} \pi_k(a_k)$
- Sample $a_{i,t}, a_{-i,t} \sim (\pi_1, \pi_2, ...\pi_N)$
- update Q tables at point $(s_t, a_{i,t}, a_{-i,t})$:

$$Q_{j,t+1}(s_t, a_t) = (1 - \alpha) Q_{j,t}(s_t, a_t) + \alpha(r_t + \gamma \Gamma_j(\pi_1, \pi_2, ...\pi_N))$$

Littman's minmax Q-learning is a simplified version of this algorithm that converges to the unique Nash Equilibrium in simple two player zero-sum games. More advanced versions of this algorithms explicitly model the opponent (permitting sub-optimal play) and multi-step ahead planning. In dynamic games where actions yeild rewards many periods in the future one can employ a Monte Carlo Tree Search to perform a forward search, simulating through own and opponent moves, and compute the bellman error. This was the technology behind AlphaZero which demonstrated superhuman performance in games like Go and Chess.

3 Literature

In this section we cover previous experimental studies concerning cooperation in games of competition and Q-learning in markets and auctions.

3.1 Iterated Prisioners' Dilemma

3.1.1 Axelrod (1984)

Many games like Cournot and Bertrand can be reduced to simpler games like the Iterated Prisioners' Dilemma (IPD). The base game has just two players having two actions (cooperate (C) or defect (D) and with four possible rewards: temptation (T), reward (R), punishment (P) and sucker (S). A game is a Prisioners' Dilemma when $T > R > P > S$. In the base game rational players are doomed to seek T and play D and end up with P; instead of playing C and getting R. In the iterated verions we often impose $(T + P)/2 > R$ so as to make defection desirable over two periods. In the IPD, played need to discount future rewards and many equilibrium are possible. To forever defect is one such Nash, but if the discount factor is high enough, we can have a sustained cooperation Nash emerge when players play strategies like Tit-for-Tat (cooperate then copy opponent) and Grim Trigger (cooperate but defect forever if it is unreciprocated). There are other cooperative Nash - but all of them are possible through the discount factor making future rewards valuable enough to prevent misbehaviour and strategies that make use of this and seek cooperation but, importantly, punish defection. One can replace infinite repititions of the game with a probability of termination and achieve the same results.

Axelrod (1980, 1984) conducted a series of computer tournaments where game theorists and computer scientists could submit programs to play in Iterated Prisioners' Dilemma. Programs would play against each other and would not know if the game would terminate or not. In these two round-robin tournaments - unambiguously Tit-for-Tat and its variants were able to accrue the largest average return accross a myriad of opponents. There were other strategies that did well but Tit-for-Tat (and its more generous variants) stood out as being the most robust against defectors, cooperators, other tit-for-tat players, other punishing strategies. It is less punishing than Grim Trigger. However, it was not effect against one opponent - that is random play (against which serial defection is best). It is also prone to defect-defect spirals against itself when mistakes are possible. In general, Axelrod's experiments showed that sustained cooperation is common - alleviating the concerns associated with traditional single period prisioners' dilemma. Axelrod highlighted the key attributes of strategies that could sustain cooperation: (1) be nice - cooperate in beginning and seek cooperation more generally (2) retaliate quickly and effectively against defection and do not be taken lightly (3) forgive opponents after retaliation and (4) be clear about your actions so as to not confuse opponents.

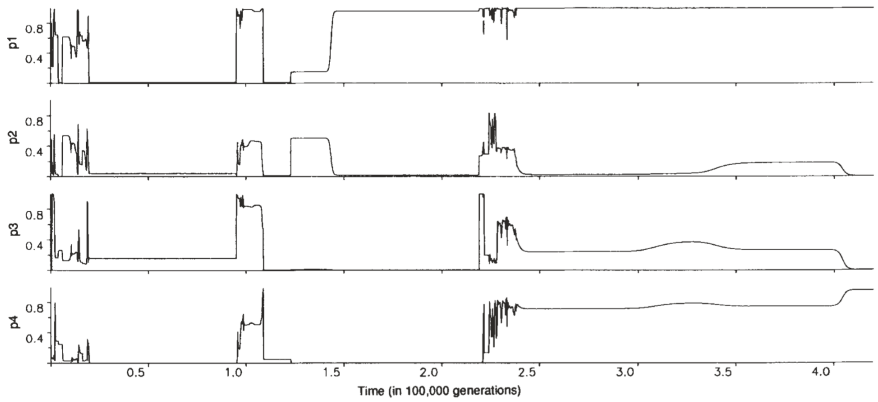
3.1.2 Nowak and Sigmund (1993)

Another series of simulational studies by Nowak and Sigmund (1993) provided another perspective on the Iterated Prisioners Dilemma if played in large populations. Their approach was an evolutionary approach where agents in a population played IPD against each other and those strategies that performed better were adopted more often. Thus successful strategies would catch on and poor strategies would not. Nowak and Sigmund (1993) studied a particular class of strategies which permitted agents to remembers one period prior in order to choose current actions. Thus actions taken today were functions of reward obtained yesterday. We have state $s_t \in \{R, S, T, P\}$ and action $a_t \in \{D, C\}$. A strategy is $a_t \sim (p_1, p_2, p_3, p_4)$ this represents the probabilities of cooperating (playing C) given $s_t \in \{R, S, T, P\}$. Based on this we can charecterise different strategies:

- Always cooperate $(1, 1, 1, 1)$
- Always defect $(0, 0, 0, 0)$
- GRIM $(1, 0, 0, 0)$
- Tit-for-tat $(1, 0, 1, 0)$
- "Pavlov" or Simpleton i.e. win-stay, lose-shift $(1, 0, 0, 1)$

The initial population has a large number of strategies proliferating. Mistakes are allowed to happen with ϵ probabilitya and high-paying policies have more offspring and catch on. And even 100 periods we see random mutations random (p_1, p_2, p_3, p_4). The results of one history of events is presented below. It

contains the evolution of the average values of (p_1, p_2, p_3, p_4) in the population. If these values are close to 1 then agents are generally cooperating and if these values are close to 1 then agents are generally defecting.



The figure shows that initially the population of myriad strategies is swamped by defectors and rampant competition ensures. However we see the emergence of a tit-for-tat (TFT) mutation that catches on and establishes cooperation. This does not last long as TFT gives way to more and more generous forms of TFT. This in turn makes the population more and more cooperative until it is vulnerable to invasion by defectors. For a while GRIM trigger catches on but gives way to the Simpleton or Pavlov. Finally, when the Pavlov strategy catches on it is immune to further invasion by other strategies. The key point is that TFT is not evolutionarily stable and is vulnerable to invasion by other strategies. Pavlov, however, is another strategy that cooperates or defects until it gets a poor reward. This strategy also performed well in Axelrod's tournament but not as well as TFT. But the main advantage with Pavlov over TFT is that it avoids defect-defect spirals with itself and most importantly, is evolutionarily stable. Thus it is robust to adoption.

The conclusion of experimental studies on the IPD tell us that not only is cooperation possible but is quite likely - there are many strategies can lead to it. The prisioners dilemma all but vanishes when we extend the game in a few more dimensions.

3.2 Experimental study in Markets

3.2.1 Waltman and Kamyak (2008)

On of the first studies to analyse Q-learning in markets was Waltman and Kaymak (2008). They used the following setup: Cournot Duopoly with Q-learning Firms. Firms took Simultaneous actions: $q \in [0, 40]$ and obtained a Period Reward $\pi = (p(\sum q) - c) * q$. In the initial runs there was no state or discounting, and so updates were made by:

$$Q_{t+1}(q) = (1-\alpha)Q_t(q) + \alpha\pi$$

Waltman and Kaymak (2008) made the important discovery that collusion can occur even without memory/discounting. And that this result holds even with many firms. This result depends on the type of exploration used (Boltzmann).

Results of computer simulations with firms that did not have a memory

		Nash	$\alpha = 0.05$	$\alpha = 0.25$	$\alpha = 0.50$	$\alpha = 1.00$
$n = 2$	Quantity	24.0	22.8 (1.3)	21.2 (1.4)	20.8 (1.2)	20.8 (1.4)
	Profit	288.0	299.1 (11.6)	312.0 (10.2)	314.7 (6.2)	314.3 (7.0)
$n = 3$	Quantity	27.0	25.1 (1.6)	22.0 (1.8)	21.5 (1.9)	22.1 (1.9)
	Profit	243.0	270.7 (22.9)	304.6 (14.5)	307.8 (14.3)	303.7 (16.6)
$n = 4$	Quantity	28.8	26.3 (1.8)	22.6 (1.9)	22.1 (2.4)	22.9 (2.6)
	Profit	207.4	252.1 (29.8)	299.0 (18.7)	301.4 (19.2)	293.2 (25.8)
$n = 5$	Quantity	30.0	27.6 (1.6)	23.2 (1.8)	22.2 (2.2)	23.3 (2.5)
	Profit	180.0	229.3 (30.0)	294.1 (17.3)	301.1 (19.2)	290.2 (28.7)
$n = 6$	Quantity	30.9	28.3 (1.5)	23.3 (2.2)	22.6 (2.6)	23.1 (3.1)
	Profit	158.7	215.4 (32.2)	290.7 (23.5)	296.3 (27.1)	289.1 (34.8)

They also offered a proof to why this may be happening. The following argument applies to reduced IPD style version of the game. $\pi_{CC}, \pi_{NN}, \pi_{CN}$ are rewards for cooperation, competition and sucker. q_C and q_N are collusive and nash quantities. In the collusive state the difference in Q-value of the two actions is $Q(q_C) - Q(q_N) \approx \pi_{CC} - \pi_{NN}$. In the competitive state it is: $Q(q_C) - Q(q_N) \approx \pi_{NN} - \pi_{CN}$. For low β (exploration parameter), $\pi_{CC} - \pi_{NN} > 2(\pi_{NN} - \pi_{CN})$ implies prob of one firm experimenting in collusive-state is lower than prob of both firms experimenting in Nash. Thus if α is high enough, the transition from Nash to Collusion needs only 1 period where both firms experiment together. Hence, firms spend more and more time in collusive-state as β falls - ultimately we see convergence to collusion.

3.2.2 Dolgopolov (2022)

We see a move fleshed out proof of the above argument in Dolgopolov (2022). We see that ϵ-greedy exploration leads to competitive equilibrium but Boltzmann exploration leads to collusion. Blue parameter region leads to cooperation using Boltzmann.

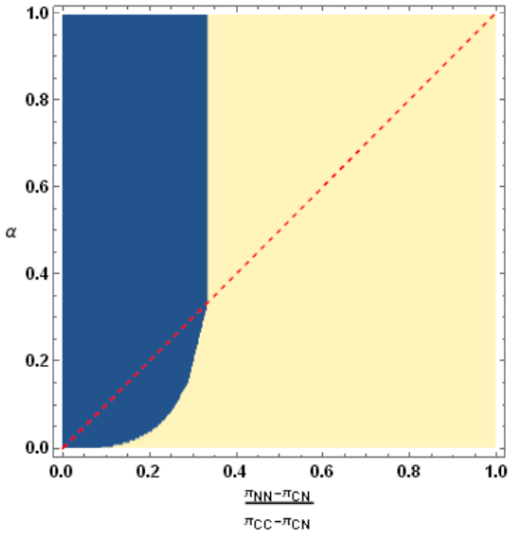


Figure 1: Learning Rate vs Reward Ratio

3.2.3 Asker et al (2022)

Asker et al (2022) studies the effect of different amount of information in the learning step. They study a repeated static Bertrand game with one competitive and one collusive Nash due to discontinuous demand. The collusive NE has higher prices/profits. Firms have no discount factor, no memory and do no exploration. The actions for firm i are $p_i \in [0, 10]$ and rewards are $\pi_i = \pi(p_i, p_{-i})$. Firm i's initial valuation is $Q^i(p) \sim UNIF(10, 20)$. At t, firm i greedily selects $p_i = argmax_p Q^i(p)$ and gets π_i. It uses this to update its Q-table by:

$$Q^i_{t+1}(p) = (1 - \alpha)Q^i_t(p) + \alpha\pi^e(p)$$

This is a very rudimentary form of Q-learning and is intended to highlight the effect of the update rule. There are two types of update rules compared. The first is an asynchronous update that happens only at $p = p_i$ and $\pi^e(p_i) = \pi_i$. This is the regular Q-learning update. The second kind of update is a synchronous update that happens at all price points $p \in [0, 10]$. Further, there are two forms of synchronous updates. The perfect information update assumes firms know the demnand function. Thus $\forall p, \pi^e(p) = \pi(p, p_{-i})$. The second update is an imperfect information update: $\forall p, \pi^e(p) = \pi_i$ if (a) $p > p_i$ and $Q_i(p) > \pi_i$ or (b) $p < p_i$ and $Q_i(p) < \pi_i$. This implies that firms only know about a "downward-sloping demand".

They find that the more the domain knowledge the firm uses in its update - the more unlikely is collusion. Asynchronous updates lead to all possible final prices, and the partial information update leads to either competitive Nash or collusive Nash. Full information updates lead to competitive Nash only.

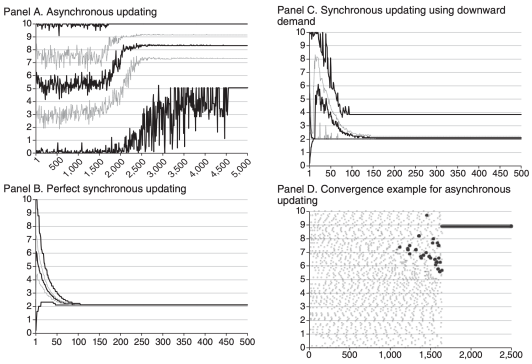


Figure 2: Prices over Periods

3.2.4 Calvano et al (2020)

Calvano et al (2020) study a bertrand duopoly with logit demand. Firms take actions p in $[p^c, p^m]$ with Boltzmann exploration. The rewards are $\pi_{it} = (p_{it} - c_{it})q(p_{it}, p_{-it})$. Each firm has a state that is a set of prices observed in the last K periods. The Bellman equation is given as:

$$V(p_-, p_-^{-i}) = \max_p \pi(p, p^{-i}) + \gamma V(p, p^{-i})$$

Average profit gain is defined as:

$$\Delta = \frac{\pi_{cnvg} - \pi_c}{\pi_m - \pi_c}$$

Where π_c and π_m are profits in Bertrand competitive equilibrium and single-firm monopoly profit respectively. They find after convergence that Δ is always above 0.5! Thus collusion is very common. It rises to 0.9 with more experimentation, slower learning, and higher discount rate. They find that these results are robust to increasing the number of firms (2 to 4), and adding cost asymmetry and uncertainty. Further, impulse response show that firms have learned to wage prices wars when one of them is forced to deviate.

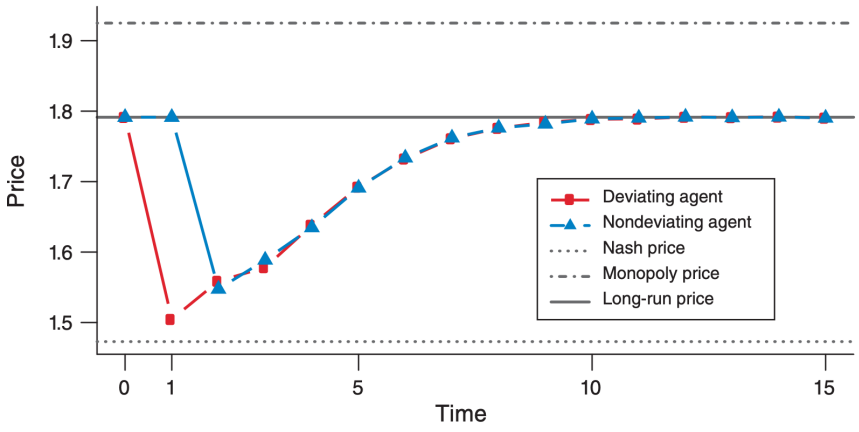


Figure 3: Impulse Response to Forced Deviation

3.2.5 Klein (2021)

Klein (2021) extends Calvano et al (2020) to sequential Bertrand duopoly (Tirole and Maskin 1988). Their model has homogenous goods, linear demand, and ϵ-greedy exploration. For firm i, state is $p_{-i,t-1}$ and actions $p_{it} \in |P|$ and value function is:

$$V(p_{-i}) = \max_{p_i} \pi(p_i, p_{-i}) + \gamma E[\pi(p_i, p'_{-i}) + \gamma V(p'_{-i})]$$

There are many theoretical equilibria: (1) Competitive Constant Prices (at or near cost) (2) Competitive Cyclical Prices: Constant undercutting and resetting. (3) Collusive Constant Prices (fear of price war). They find that for smaller $|P|$ we get (3) is more likely and for higher $|P|$ we get (2) is more likely. The first figure contains impulse responses for (3) and the second figure depicts the converged prices accross games in (2).

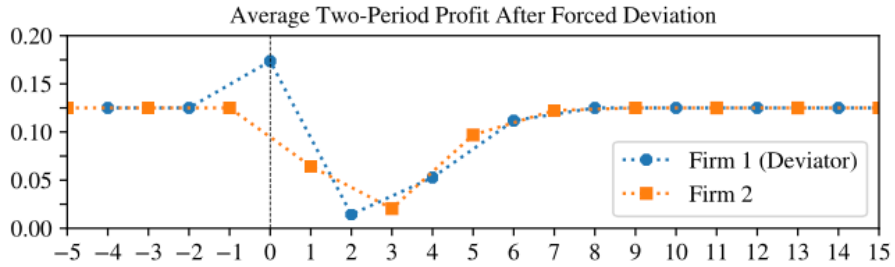


Figure 4: Smaller $|P|$ - Collusion

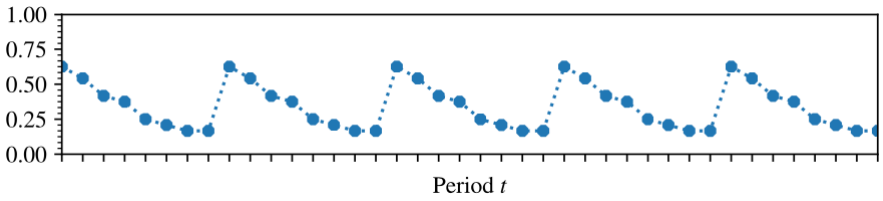


Figure 5: Higher $|P|$ - Competition

3.2.6 Banchio and Skrzypacz (2022)

Banchio and Skrzypacz (2022) extend the previous studies to repeated first price auction (FPA) and second price auction (SPA). They consider two bidders that can take actions $b_i \in \{b_1, b_2 ... b_m = 1\}$ with valuation $v_i = 1$. The rewards $v_i - b_{-i}$ (SPA) and $v_i - b_i$ (FPA). The theoretical competitive Nash equilibrium has both players play b_m (FPA/SPA) or both play b_{m-1} (SPA). If discount γ high, we can get collusive Nash equilibrium as well. There are two kinds: (1) strongly symmetric - where bidders play b_1 until deviation. (2) Bid Rotation - where bidders rotate between b_1/b_2 (FPA) or b_1/b_m (SPA) until deviation. After deviation they revert back to Competitive Nash forever. They simulate the games without state and find that SPA leads to competitive Nash while we can see collusion in FPA. Further, revealing bids (and allowing for a state) and synchronous updates (like Asker et al 2021) restores competition even in FPA. The figure below shows how FPA leads to collusion while SPA leads to cooperation.

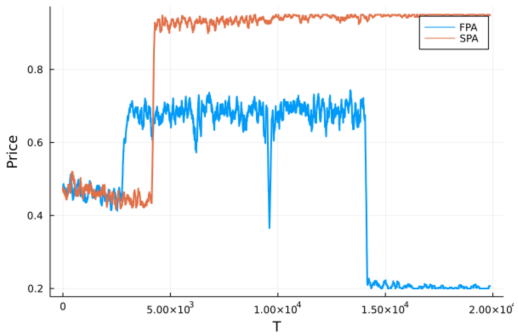
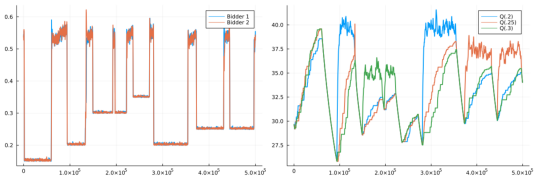


Figure 6: Bids vs Periods

Further they find that if exploration never ends then bidders may never converge - they keep moving. There is temporary stability when both bids are the same, but this is vulnerable to exploration.



(a) Bids (b) Q-values

Figure 7: Bids vs Periods

3.2.7 Zhang (2022)

Zhang (2022) studies Deep Q-learning Networks in a repeated Bertrand duopoly. He tests for different kinds of networks - Deep Neural Network (DNN), Recurrent Neural Network (RNN), Long Short Memory Network (LSTM). Uniformly, the results show that competition is the norm. The author conjectures that this is due to "Experience replay" - where large random samples of $(s_t, b_t, s_{t+1}, b_{t+1})$ are drawn uniformly from a cache to form a loss function to update network parameters. This sampling eliminates temporal correlations in data used to learn. The average profit above competetive levels in all simulations goes to zero.

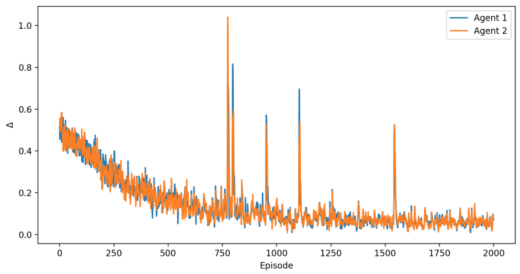


Figure 8: Avg Bids vs Periods

4 Simulations

In this section I present my own simulations. I cover Bertrand, Cournot, first and second price auctions. I replicate most of the results in the literature and add a few of my own.

4.1 Pricing Games

In this section we study what helps or hinders algorithmic collusion when Q-learning pricing bots interact in the market. We will consider the effect of seven factors. First are the algorithms themselves - exploratory strategies, discount

factor, and the mode of learning. The second is feedback or the role of the state. Here we consider two types of feedback: past prices and reputation. We also consider some changes in the environment, in particular the number of competitors and seasonal demand.

4.1.1 The Bertrand Pricing Game

The experimental setup is as follows. Two firms compete in a differentiated products pricing game with demand given as:

$$D(p, p') = \frac{e^{(a_1 - p)/\mu}}{e^{(a_1 - p)/\mu} + e^{(a_2 - p')/\mu} + e^{a_0/\mu}}$$

Period profit is given by,

$$\pi(p, p') = (p - c)D(p, p')$$

The parameters are: a_i quality of outside, own and opponents' good; μ being product own-price sensitivity; c constant marginal cost. For all experiments we choose the following parameters:

$$a_0 = 2, a_1 = 20, a_2 = 20, \mu = 0.1, c = 0$$

This leads to the following stage game rewards:

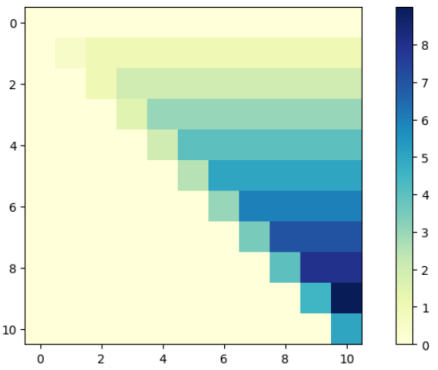


Figure 9: Stage Game Rewards for Firm 1

4.1.2 Q-Learning

In Q-learning terms the actions are prices $p_t \in \{0, 1, 2, 3, 4...10\}$ and s_t can be opponents' past price p'_{t-1} or some other data. A policy $\sigma(s_t)$ is a strategy of choosing actions given states. The discount factor is γ. For policy σ, every state s has a **expected return**, assuming the opponents plays by σ'.

$$V_{\sigma,\sigma'}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t \pi(p_t, p'_t) \mid s_0 = s, \sigma, \sigma'\right]$$

$$Q_{\sigma,\sigma'}(s,p) = \pi(p,p') + E\left[\sum_{t=1}^{\infty} \gamma^t \pi(p_t, p'_t) \mid \sigma, \sigma'\right]$$

We make use of the concept of an "Experience based equilibrium". It is $(\bar{\sigma}, \bar{\sigma'})$ arrived at by an iterative process. One such process is Multi-agent Q-learning:

- Guess $Q_0(s,p)$
- at t, do:
  - observe s_t
  - take action p_t from **exploratory strategy**
  - collect reward π_t
  - observe transition s_{t+1}
  - update Q at point (s_t, p_t):

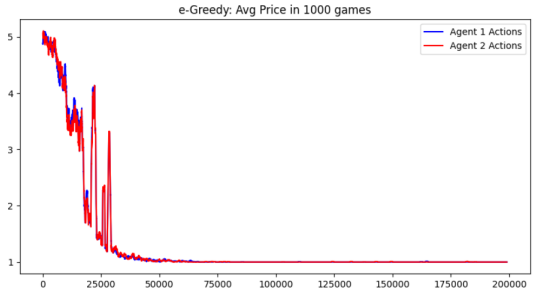
$$Q_{t+1}(s_t,p_t) = (1-\alpha)Q_t(s_t,p_t) + \alpha(\pi_t + \gamma max_{p'} Q_t(s_{t+1},p'))$$

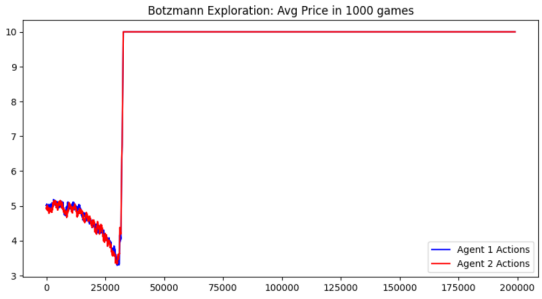
  - Learning rate α: weight given to current experiences over past

As bots' Q tables stabilize, $\bar{\sigma}(s) = argmax_p Q(s,p)$ is going to be optimal against $\bar{\sigma'}(s)$ and both will be best responses to each other.

4.1.3 Simulation 1: Exploration

The first simulation is to demonstrate the effect of different exploration strategies on collusion. We have no state or discounting and so we set $\gamma = 0$, $\alpha = 0.3$. We find that Boltzmann Exploration leads to collusion while ϵ-Greedy does not.

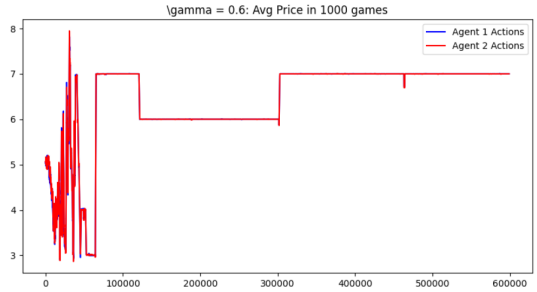


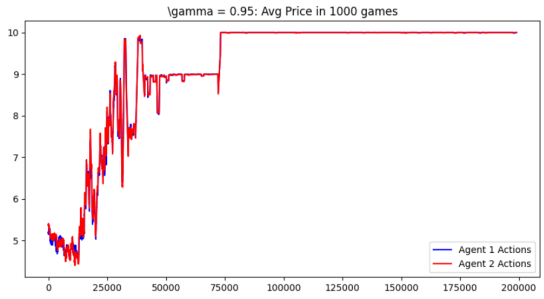


This result supports the conclusion in Waltman et al 2008, Dolgopolov 2022. The probability of exploration is more in lower profit regimes as compared to higher, which is a feature of Boltzmann exploration, gives rise to the possibility of agents simultaneously discovering collusion and then afterwords not exploring as much.

4.1.4 Simulation 2: Discounting

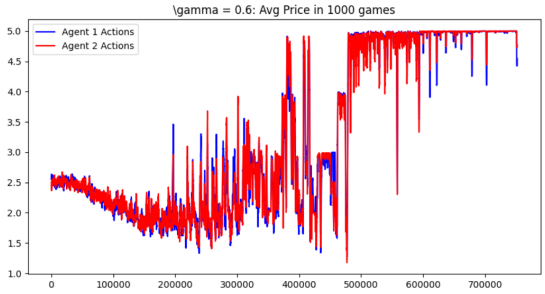
In the second simulation we introduce the effect of discounting. This introduces a strong force towards collusion. We retain all parameters the same as before. Thus there is no state, $\alpha = 0.3$ and we use ϵ-Greedy exploration (which previously, with no discounting led to competition). The results show that discounting has a strong effect on the final result - which is now collusion. This confirms the basic idea in economic theory that a high discount rate provides impetus for collusion (Ivaldi et al 2003).

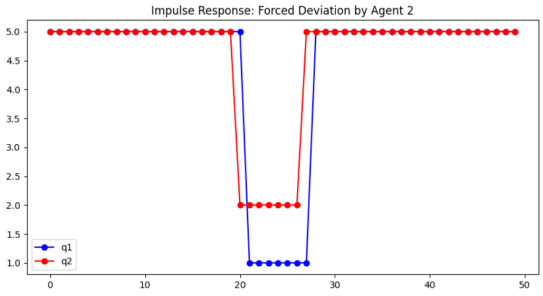




4.1.5 Simulation 3: Memory

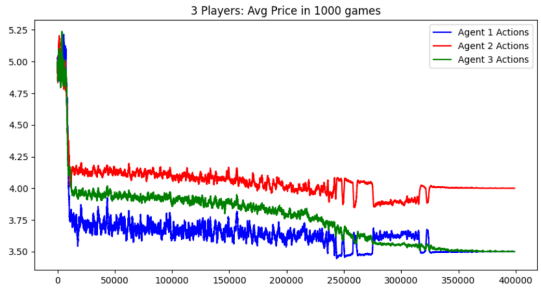
In the third simulation we consider the effect of memory. We retain the same parameters as simulation 2 but introduce a single state - the past prices. Thus the state is p'_{t-1}, $\gamma = 0.95$, $\alpha = 0.3$ and we use ϵ-Greedy exploration. The result is the same as before - collusion occures - but a closer look shows that the agents are using the memory thus provided in manufacturing it. If we look at impulse responses to a forced deviation from equilibrium path (a unilateral defection) we find that the opponent is responding by retaliation. Thus we see that introducing a state allows agents to play strategies that are retaliatory, and this supports collusion. This goes with the conclusion of Axelrod 1980 and Calvano et al 2020 that the threat of retaliation via memory supports collusion.

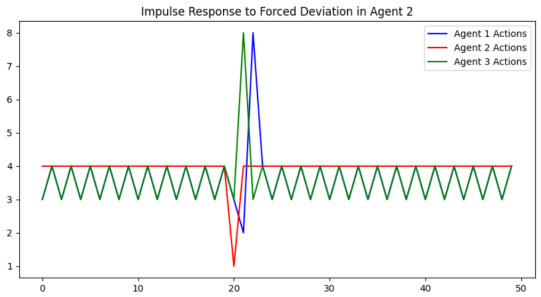




4.1.6 Simulation 4: Competitors

In the fourth simulation we consider the effect of new competitors. We find that now each firm must track the past prices of previous firms. State is (p'_{t-1}, p''_{t-1}) and the other parameters are: $\gamma = 0.95$, $\alpha = 0.1$, ϵ-Greedy exploration, $\mu = 2$. We find that collusion is still possible - albeit harder to attain. Impulse responses confirm that retaliation happens.

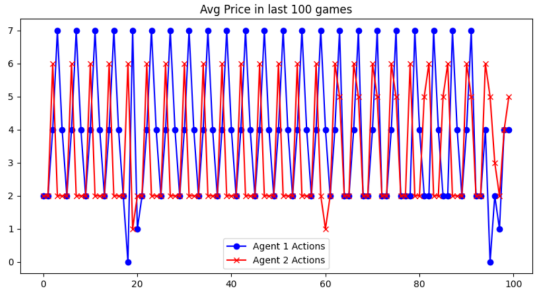


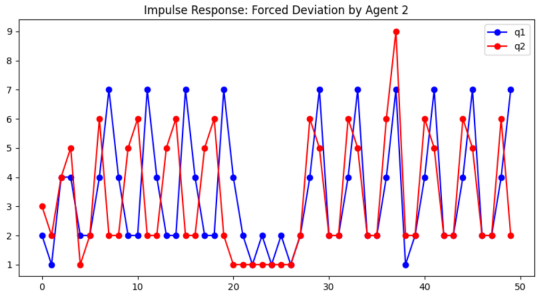


4.1.7 Simulation 5: Edgeworth Cycles

In the fifth simulation we consider the effect of seasonal or cyclical variation in demand. This simulation is just a simple extension of the base result - but it demonstrates interesting dynamics. We introduce a temporary reduction in demand every even period, which is restored in the odd periods.

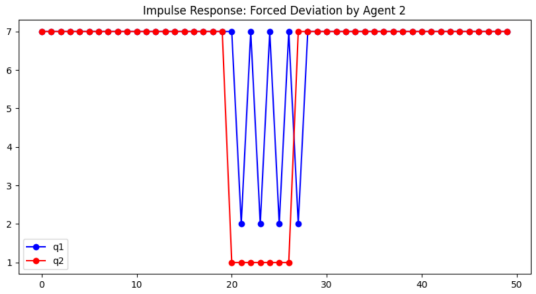
State p'_{t-1}, discounting $\gamma = 0.95$, $\alpha = 0.3$ and ϵ-Greedy exploration.

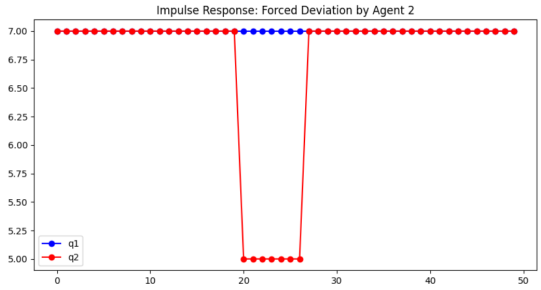




4.1.8 Simulation 6: Reputation

We use a new type of state in this experiment. State is R'_{t-1}, which is set to 0 if opponent prices 2 units below, else 1. Rest of the parameters are: $\gamma = 0.95$, $\alpha = 0.3$, ϵ-Greedy exploration. Reputation simplifies tracking others' actions (Nowak 2006) and provides a simple mechanism permit retaliation. Impulse responses show that retaliation only happens when the opponents misbehaves sufficiently to lose repute.





4.1.9 Simulation 7: Modes of Learning

In the final simulation we consider the effect of alternative modes of learning. In traditional Q-learning we see that each state-action pair is updated one at a time. This means that firms are not computing counterfactual profits at any other price points - using any heuristic. If we relax this assumption and allow firms to know the demand function and compute profit accross all state-action pairs we see a dramatic change. As both firms incorporation greater restrictions about what their opponent will do, Q-learning convergence quickly to Nash-competitive levels. We have no state, no discounting, $\alpha = 0.3$, and Boltzmann exploration. These are conditions ripe for collusion. We let firms know reward table, so can update entire Q-table at once:

$$Q_{t+1} = (1 - \alpha)Q_t + \alpha\pi(., p'_t)$$

We find that Q-learning converges to competition in the presence of full information updating. In contrast, we see collusion when we use the traditional Q-learning algorithm.

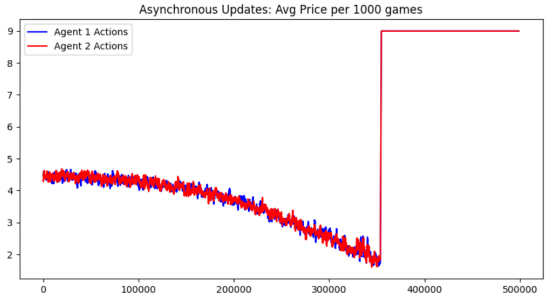


Figure 10: Asynchronous Update

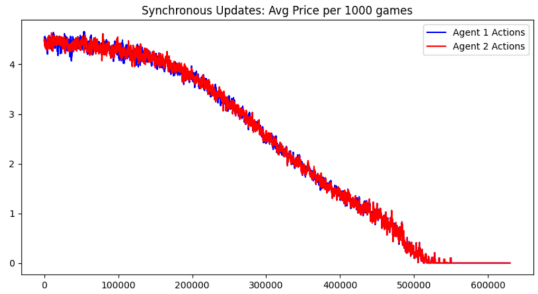


Figure 11: Synchronous Update

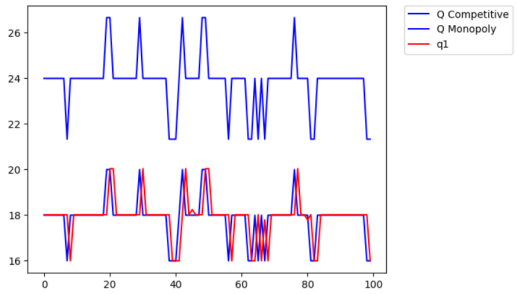
4.2 Quantity Games

4.2.1 Cournot Duopoly

We study a Cournot Duopoly with stochastic demand. The setup is the same as Waltman and Kayak (2008). We use Q-learning with Boltzmann Exploration. Demand is given by $P = u_t - v\sum_i q_i$. Here u_t is stochastic and given by $u_t = 40 + e_t$ where $e_t \sim \{-4, 0, 4\}$ with known transition matrix P. In this example (u_{t-1}, q_{t-1}) is the state (after each game firms learn about demand and opponents' quantity). And q_t is the action. In this model we make an implicit assumption that firms know P the transition matrix and thus can compute the expected maximum on the RHS of the Bellman equation.

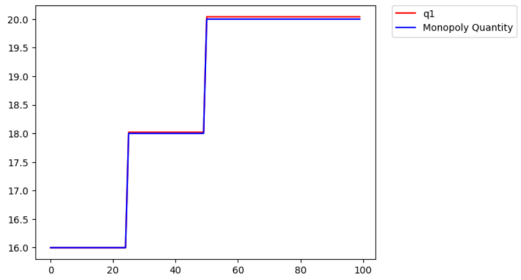
4.2.2 Simulation 8: Stochastic Demand

In this section we only permit a single firm and we want to see if this firm can correctly discover monopoly prices via Q-learning. Since the environment is stochastic, so are monopoly prices. We find that Q-learning finds the monopoly price and picks up the entire consumer surplus. The figure below represents the last 100 games played.



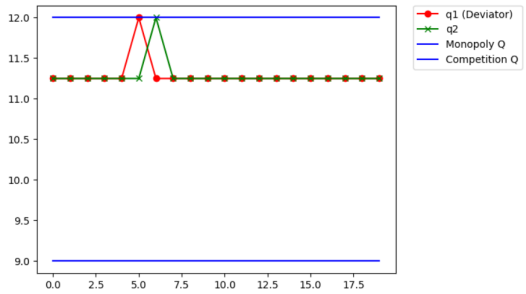
We also look at the impulse responses to changing seasons. Here we force the demand to be low, medium and high in a staggered fashion. We see that

the bot is able to respond to past information and correctly find the monopoly price.



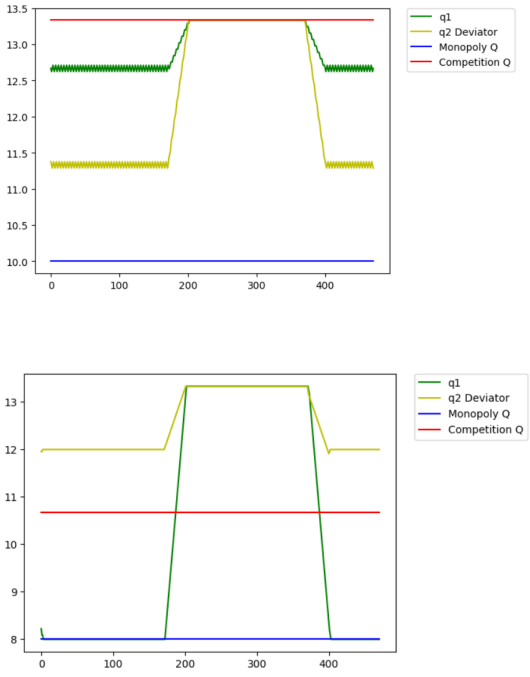
4.2.3 Simulation 9: Retaliation

In this simulation we switch off the stochasticity and just study the simple cournot duopoly. Thus demand does not change but firms register their opponents' moves. We find that firms are able to produce below competitive levels. They are able to support this equilibria through the threat of retatiation (quantity war). The impulse response to a single-period forced deviation are shown below.



4.2.4 Simulation 10: Stochastic Demand and Retaliation

In this simulation we test the full specification. We allow both stochastic demand and study the duopoly case. We set $e_t \in \{-4, +4\}$ with equal probability. We find that firms are again able to support non-competitive quantities. However, the results are more ambiguous - one firm begins to produce much higher quantity in bad times (low demand) while in good times firms are colluding. Still, impulse responses show evidence of quantity wars.



4.3 Bidding Game

In this section we consider if Q-learning bots learn to suppress bids in auctions. We begin with a comparision of the first price vs second price auction with sealed bids. Then we study the situation where the previous bids are revealed. Lastly we look at the effect of Deep Q-learning.

4.3.1 First and Second Price Auctions

We consider the case of the repeated static auctions.The stage game is as follows: Two bidders compete in an auction with bids: $b \in \{0, 0.1, 0.2...1.0\}$. They have a common valuation $v = 1$. The period profit function depends on the nature of the auction. In the first price auction we have: $v - b'$ if $b \geq b'$, $0.5(v - b')$ if $b = b'$, else 0. In the second price auction we have: $v - b$ if $b \geq b'$, $0.5(v - b)$ if $b = b'$, else 0. The state is s is opponents' past bid b'_{t-1} and a policy $\sigma(s)$ is a strategy of choosing bids given states. The discount factor is γ. Given this structure, the rewards are as follows. The x-axis is firm i's action and y-axis is firm -i's action.

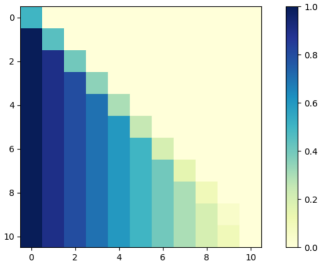


Figure 12: Second Price Rewards

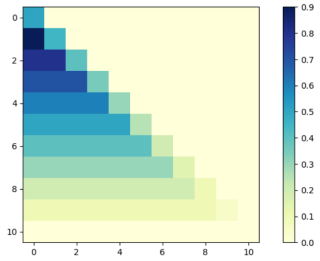


Figure 13: First Price Rewards

4.3.2 Q-Learning

For policy σ, every state s has a **expected return**, assuming the opponents plays by σ'.

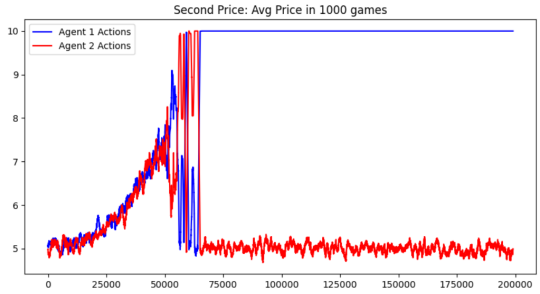
$$V_{\sigma,\sigma'}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t \pi(b_t, b'_t) \mid s_0 = s, \sigma, \sigma'\right]$$

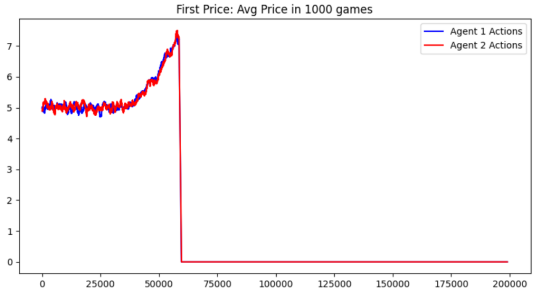
$$Q_{\sigma,\sigma'}(s,b) = \pi(b,b') + E\left[\sum_{t=1}^{\infty} \gamma^t \pi(b_t, b'_t) \mid \sigma, \sigma'\right]$$

We again employ the concept of "Experience based equilibrium". This is $(\bar{\sigma}, \bar{\sigma'})$ arrived at by an iterative process. One such process is Multi-agent Q-learning. As bots' Q tables stabilize, $\bar{\sigma}(s) = argmax_p Q(s,b)$ is going to be optimal against $\bar{\sigma'}(s)$ and both will be best responses to each other.

4.3.3 Simulation 11: First vs Second Price

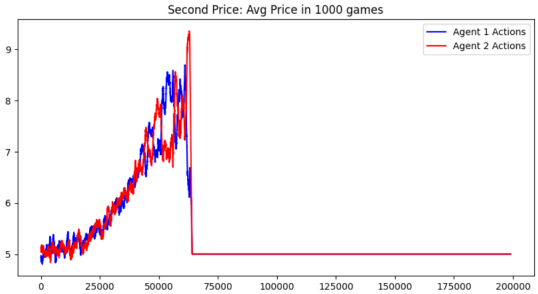
In this simulation we study the effect of different auction rules on collusion. We have no state or discounting $\gamma = 0$, and set $\alpha = 0.5$. We find that the first price auction leads to collusion, while second price auction remains robust and attains competitive outcomes.

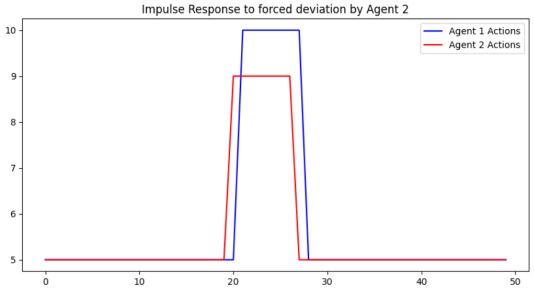




4.3.4 Simulation 12: Revealed Bids

In this section we consider the role of the state when bids are revealed. Thus bots know b'_{t-1} and have discounting $\gamma = 95$. We set $\alpha = 0.5$. With help of a memory, we find that even in the second price auction we see collusion.





4.3.5 Simulation 13: Deep Q-Learning

We tested Deep Q-learning networks and got the same result as Zhang (2022) - in both FPA and SPA, with and without a state we were unable to generate collusion. This could possibly be due to uniform experience replay which samples uniformly from cache to update the network parameters - and thus kills the temporal correlation in the data.

5 Conclusion

5.1 Overview

As the world becomes increasingly algorithmized there is a need to delve deeper how these algorithms behave, interact and learn. There is a need to study the existing rules - which were designed for human interaction - and if they stand up to scrutiy when applied to algorithmic interactions. New technologies always come with disruption and unrest and making progress in understanding dynamics of algorithmic interaction will help soften the impact and prevent marginalization of weaker participants in the market and corrosion of the health of market institutions.

What have we learned from all these studies? First, that market design really matters - the rules govern the final outcomes. Second, it seems that cooperation can establish itself very easily. The real question should be about how we can inject competition in settings where cooperation is easily possible. Third, it is also evident that specifics of algorithms themselves are critical determinants of the final outcomes. Fourthly, market design for adaptive agents are markedly different from that for rational agents. The iterative process of finding an equilibria matters - bots have no way to jump to the correct and most rational solution.

There is now a sizable body of literature documenting the potential pitfalls of algorithms. But considerable work needs to be done. On the empirical front - a lot of data has to be studied to detect the behaviour of actually existing algorithms. On the theoretical front - we can draw on tools from evolutionary game theory and mean field games to map hyperparameters to outcomes. Mechanism design and market design can be used to study the impact of these algorithms in different markets. And in areas where we are unable to model and we have limited data - simulations can throw light.

There is also a need to overcome the bridge between academics who engage in theoretical research and practitioners in industry who employ black-box methods. The academics have clean results in stylized settings while the practitioners can conjure high performance in complex environments. Both gain from cross-fertilization.

5.2 Open Questions

In this section we consider some open areas of research in this topic.

First, is the study of collusion in games with delayed rewards - such as the English or Dutch auction. To impliment reinforcement learning in such games is a bit more involved because we need to impliment eligibility traces so that sparse rewards can be connected to the state-action pairs that led to it. Such auctions permit a wider variety of strategies - bid spipping, bid increments, etc.

Second, papers usually study Q-learning but not direct policy learning. This is important because policy learning can lead to the possibility of mixed strategie being optimal responses. Thus a mixed strategy collusive equilibria becomes

possible in such a learning algorithm.

Thirdly, as well have seen Deep Q-learning does not show signs of supporting collusion. This is chiefly due to the uniform sampling method in its update step. As we know from our study of Boltzmann exploration, recent experiences are critical in supporting collusion. A straightforward implimentation would be to study Deep Q-learning networks with prioritized experience replay. By choosing the samples wisely to train networks - bots should be able to generate collusion.

Fourthly, the role of exploration has been demonstrated to be important in supporting collusion. So far we have only looked at simple ϵ-greedy or Boltzmann exploration. But there are other ways: goal-directed exploration, diversity-of-action as virtue, exploration with upper confidence bounds, noise-based exploration and human-expert guided exploration.

Fifthly, as Nowak (2006) highlights there are many ways that natural selection can lead to collusion. This involves a wide variety of mechanisms: kinship, direct and indirect recipocity, networks and group formation. To build a master theory of cooperation-competition in very general setting could also prove useful. This includes the forces of reputation and possibility of communication.

Sixthly, as Asker (2021) shows - if bots have even a small heuristic about the environment or their opponents' behaviour; the final outcome can change dramatically. This is also important because it is very likely that firms will not go in blind (beyond Bandit like experiements and A/B tests) and will add layers of intelligence which incorporate domain expertise to their bots. Incorporating knowledge about the environement is model-based learning. We already know that Q-learning is too slow. There needs to be ways for managers to pass on best practises to Q-learning agents. Of course, the model provided could be imperfect or even flat out wrong. But this seems an important extension to explore.

Sevently, very few studies have studied more complex demand systems. Simple extensions can be to study collusion with multiple products, personalized pricing, forward-looking consumers, network effects and "lock-ins", brand loyalty and buyer power.