Deep Q-learning of prices in oligopolies: the number of competitors matters

Herbert Dawid* Philipp Harting† Michal Neugart‡

August 2024

Abstract

Artificial intelligence algorithms are increasingly used for online pricing and are seen as a major threat to competitive markets. We show that if firms use a deep Q-network (DQN) as an example of a state-of-the-art machine learning algorithm, prices are supra-competitive in duopoly but quickly move to competitive prices as the number of competitors in an oligopoly increases. This finding is very robust concerning variations of the exploration and learning rate used in the DQN algorithm.

Keywords: algorithmic price setting, deep Q-network, oligopoly, supra-competitive prices

1 Introduction

Firms increasingly turn to algorithms to support their decision processes and autonomously determine their actions. This is particularly true for transactions taking place on online platforms. It is conjectured that retailers track their competitors' pricing decisions and that about "Two thirds of them use software programs that autonomously adjust their own prices based on the observed prices of competitors". (European Commission, 2017, p. 5).

*Department of Business Administration and Economics and Center for Mathematical Economics, Bielefeld University, Germany, hdawid@uni-bielefeld.de

†GREDEG, Université Côte d'Azur, France and Bielefeld University, Germany, philipp.harting@uni-bielefeld.de

‡Department of Law and Economics, Technical University of Darmstadt, Germany, michael.neugart@tu-darmstadt.de

Concerns have been raised, among others, by competition authorities that "The availability of real-time pricing information may also trigger automatised price coordination. The wide-scale use of such software may in some situations, depending on the market conditions, raise competition concerns." (European Commission, 2017, p. 5), see also (Competition and Market Authority, 2018; Monopolies Commission, 2018).

Following up on an earlier contribution by Waltman and Kaymak (2008), the concerns about algorithmic collusion have been reinforced by several recent analyses showing, in simulation models, the emergence of supra-competitive prices of firms in duopolies with different market designs when (tabular) Q-learning algorithms (see, e.g. Calvano et al., 2020, 2021; Klein, 2021; Abada and Lambin, 2023; Epivent and Lambin, 2024) or reinforcement learning with synchronous updating (Asker et al., 2022, 2024) are used for pricing. The amount of collusion is, however, mitigated or even overall eliminated if the algorithm can incorporate additional information about a market's demand structure, referred to asynchronous updating (Asker et al., 2022, 2024).

Which pricing algorithms firms use is a black box, such that it is of great importance to understand the robustness of these results concerning the type of algorithms employed by firms. As it has been used in market simulations so far, basic tabular Q-learning algorithms were mostly developed in the 1990s. Importantly, they are not designed to deal with large state spaces, which arise as firms choose a particular price from a possibly large price vector. Therefore, as is also conjectured by competition authorities (Competition and Market Authority, 2018), it seems natural that firms use more advanced algorithms rather.

This paper explores the possibility of supra-competitive prices in an oligopoly where the pricing algorithm is a deep Q-network (DQN). Considering the interaction of firms whose pricing decisions are taken by DQNs allows us to address the important question of how the number of competitors in a market populated by firms relying on algorithmic pricing influences the level of emerging prices.

Deep Q-network algorithms are not only prevalent in the machine learning literature (Mnih et al., 2015; Goodfellow et al., 2016), but also are heavily used in the recent applied literature on solving complex firm problems (Liu, 2023; Zhao and Lee, 2022; Qiu et al., 2020). Compared to tabular Q-learning, DQNs exhibit faster convergence and, more importantly, do not suffer from the curse of dimensionality if the dimension of the state space grows. This is of great importance in market settings where the state is often given by the vector of actions of all competitors in the previous pe-

riod, which implies that the number of firms in the market determines the dimension of the state space. While this strongly restricts the number of interacting firms that can be handled using tabular Q-learning, the use of DQNs also allows firms to learn pricing strategies in markets with many competitors.[1]

Our main findings are that, whereas in a duopoly setting DQNs might lead to prices very close to the monopoly level, the supra-competitive pricing resulting from the interaction of these algorithms quickly vanishes as the number of firms increases. In particular, markets with at least five competitors robustly yield prices close to the Nash equilibrium level. These findings are robust concerning changes in the key parameters of the algorithm.

2 Deep Q-learning in oligopolies

To closely link our contribution to previous analyses, our market setting, in which firms use DQNs to set prices, is equivalent to Calvano et al. (2020). It is a repeated pricing game in which firms face a logit demand

$$q_{i,t} = \frac{e^{\frac{a_i - p_{i,t}}{\mu}}}{\sum_{j=1}^{k} e^{\frac{a_j - p_{i,t}}{\mu}} + e^{\frac{a_0}{\mu}}}, \tag{1}$$

where a_i are product quality indexes, a_0 the attractiveness of an outside good, k the number of firms, and $\mu > 0$ an index governing the degree of horizontal differentiation. The price of firm i in period t is $p_{i,t}$, and the marginal cost is denoted by c_i, such that firms maximize profits $\pi_{i,t} = q_{i,t}(p_{i,t} - c_i)$. We denote the Nash equilibrium price with p^N and the price under perfect collusion (i.e., maximizing the sum of all profits) with p^M, which we refer to as the monopoly price. We set all market parameters to the values used in (Calvano et al., 2020) (see Online Appendix A).

Firms set prices simultaneously and a DQN algorithm decides every period on a firm's price.[2] More specifically, the algorithm sets its price using a deep neural network which represents an action-value function $Q(s_i, p_i; \theta_i)$, where $s_i \in S$ is the state of firm i in period t, p_i a potential action of firm i in period t, i.e., a price from a finite set of prices P, and θ_i the

[1] Tabular Q-learning in (Cournot-)oligopolies with up to six firms has been studied in Waltman and Kaymak (2008), however, they assume that firms, apart from their action, only take into account the average action of all competitors, such that the dimension of the state space is independent from the number of firms.

[2] A detailed description of the DQN, including a pseudocode and the values of all parameters, is provided in the Online Appendix A.

vector of weights in the neural network. The state space S is given by all possible combinations of prices chosen by all firms in the previous period, $s_{i,t} = (p_{1,t-1}, .., p_{k,t-1})$.

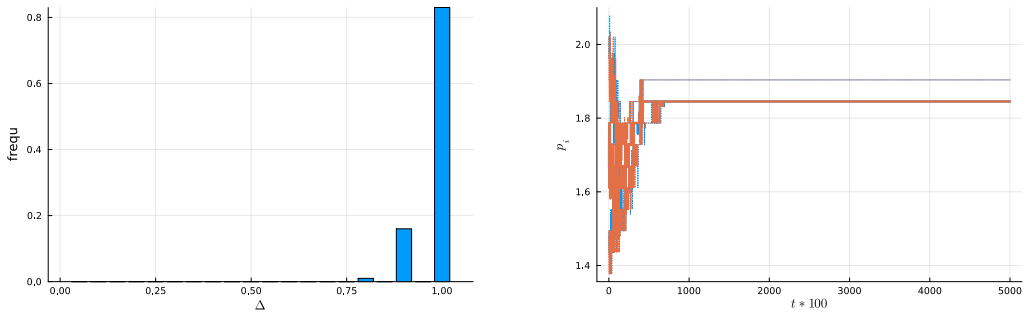
The input nodes of a DQN, which represent the action-value function $Q(s_i, p_i; \theta_i)$, correspond to the components of the state s_i of firm i. Each output node corresponds to a price $p \in P$ and estimates the Q-value for this price under the current state. In our DQNs, there are two hidden layers between the input and the output layer, with the number of nodes in each hidden layer following rules of thumb developed in the literature on deep neural networks (see, e.g., Bengio, 2012)

Firm i trains the DQN by adjusting every period the vector of weights $\theta_{i,t}$ to reduce the mean-squared error of the right-hand side of a Bellman equation. The adjustment of the weights follows a standard gradient descent algorithm with a learning rate α. As a default in our analysis, we use the algorithm with '*online gradient descent*' (Bengio, 2012), which means that updating is based only on the most recent observation. Alternatively, the updating of weights might be based on a random sample from a backward window of past observations, referred to as '*experience replay*' (see, Mnih et al., 2015). Our motivation to mainly use online gradient descent is to stay close to the existing papers on Q-learning on markets (e.g. Calvano et al., 2020; Epivent and Lambin, 2024), which all rely only on the most recent observation in each step of their training step.[3] However, we check the robustness of our findings concerning the use of experience replay and provide the corresponding results in the Online Appendix B.

A greedy algorithm guarantees that the actions chosen are experimented with. The ϵ-greedy model of exploration chooses the action $p_{i,t}$ maximizing $Q(s_{i,t}, p_i; \theta_{i,t})$ with probability $1 - \epsilon_t$ and randomizes uniformly over all actions with probability ϵ_t. The exploration rate declines with $\epsilon_t = e^{-\beta t}$, where $\beta > 0$ is the exploration parameter. Thus, initially, firms choose their prices to a large extent randomly, but as time passes, actions in line with the highest Q-value become more likely.

In each simulation run, we let the firms train their DQNs for $T = 5 \cdot 10^5$ periods while interacting with each other. We then study the long-run prices and profits emerging if each firm in each period chooses the price $p = \arg\max_p [Q(s_t, p; \theta_i^*)]$, where θ_i^* denotes the network weights learned

[3] In other game theoretic frameworks, such as the prisoner's dilemma, recent work has, however, explored properties of reinforcement learning of tabular Q type with training data drawn from backward windows with size substantially larger than one, see, e.g.,Barfuss and Meylahn (2023).



(a) (b)

Figure 1: Panel (a) shows the distribution of long-run values of profit gain Δ and panel (b) dynamics of mean values and 25% as well as 75% quantiles (dotted) of prices of firm 1 (blue) and firm 2 (yellow) in a duopoly with DQN parameters $\alpha = 0.001, \beta = 6 \cdot 10^{-5}$.

by firm i at the end of the T training periods.[4] If prices converge to a fixed point, the long-run profit $\bar{\pi}_i$ for firm i is given by the firm's profit in this fixed point. If price cycles emerge, the average profit of a firm during the cycle determines $\bar{\pi}_i$. Following the recent literature in this area (c.f. Calvano et al., 2020), we use the profit gain $\Delta = (\bar{\pi} - \pi^N)/(\pi^M - \pi^N)$ with $\bar{\pi} = \frac{1}{k}\sum_{i=1}^{k} \bar{\pi}_i$ as the indicator for the degree of supra-competitive prices emerging in the market.

3 Results

We first address whether DQN learning in duopoly markets generates supra-competitive long-run prices, as observed in simulations relying on tabular Q-learning algorithms. Figure 1 depicts the distribution of long-run values of Δ across 100 batch runs as well as the corresponding dynamics of mean prices for a duopoly ($k = 2$), with a learning rate $\alpha = 0.001$ and an exploration parameter $\beta = 6 \cdot 10^{-5}$.[5] The figure shows that for this parametrization of the DQN algorithm, both firms set prices at or at least very close to the monopoly level in all runs. The mean value of the profit gain across the runs

[4] All presented results are based on a Julia implementation of the model using the Flux package for implementing the deep neural networks and the gradient descent algorithm.

[5] It should also be noted that the value of the learning rate α is not directly comparable to the learning rates reported in papers using tabular Q-learning, since the interpretation of this parameter in the context of the stochastic gradient descent used in DQNs is different.

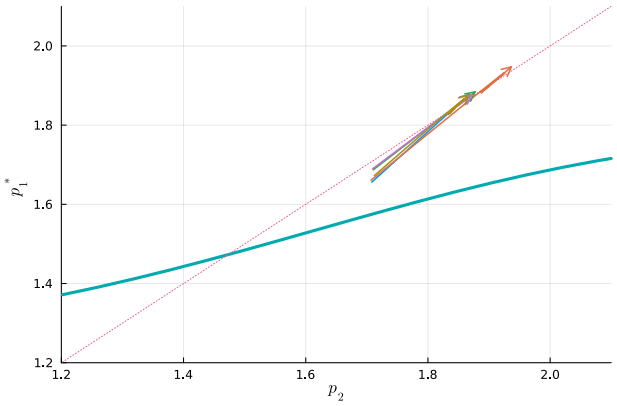


Figure 2: Dynamics of greedy prices of firm 1 as functions of the competitor's price in the previous period for five single runs in a duopoly with DQN parameters $\alpha = 0.001, \beta = 6 \cdot 10^{-5}$. For each run, lines are generated by estimating third-order polynomial functions $f_1(t)$ and $f_2(t)$ using the time series of $p_2(t-1)$ and $p_1^*(t)$ and plotting $(f_1(t), f_2(t)), t = 0, .., T$ with the arrow indicating the direction as time increases. The purple line shows the best reply function of the (static) duopoly model. It intersects the red line at the Nash price $p^N \approx 1.47$.

in the batch is $\Delta = 0.96$, which is substantially higher than the Δ values reported for tabular Q-learning in Calvano et al. (2020).[6] Furthermore, convergence typically occurs after about 80.000 periods and, therefore, is about ten times faster than results based on tabular Q-learning reported in the literature.

To illustrate the mechanism underlying the emergence of supra-competitive prices, Figure 2 shows, for five single runs, how greedy prices of firm 1 ($p_1^*(t)$) relate to the price of firm 2 in the previous period ($p_2(t-1)$). The figure clearly shows that DQN does not learn to follow the best reply function of the (static) duopoly model but instead seems to mimic price increases of the competitor with a slight delay such that a coordinated, gradual ascent towards the monopoly price emerges, as can be seen between periods 40.000 and 80.000 in Figure 1(b).

Having established that strongly supra-competitive prices might emerge in duopolies with pricing decisions made by DQN algorithms, we now exam-

[6]Depending on the choice of the learning rate α and the exploration parameter β Calvano et al. (2020) obtain average profit gain values between 0.65 and 0.9.

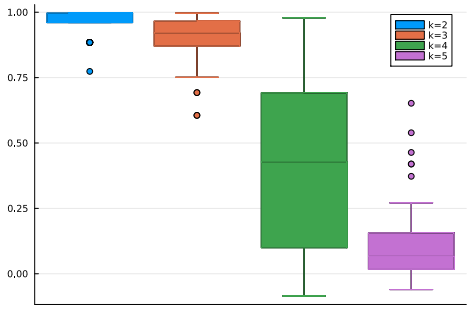


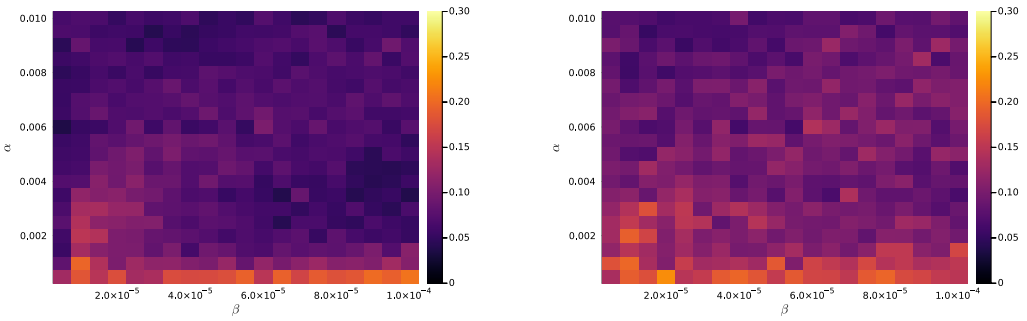
Figure 3: Boxplots of distribution of long-run values of profit gain Δ for $k = 2, 3, 4, 5$ firms and $\alpha = 0.001, \beta = 6 \cdot 10^{-5}$.

ine how robust this phenomenon is concerning an increase in the number of competitors. In Figure 3, we show boxplots of the long-run prices over 100 batch runs for the duopoly ($k = 2$) and oligopolies with three to five firms under our standard parameter setting. While the profit gain is still close to one for $k = 3$, it quickly decreases as more firms are in the market. For $k = 4$ firms, the distribution of emerging long-run prices is very broad. Examining the exact distribution of Δ values (not shown here) reveals that whereas prices in the direct neighborhood of the Nash equilibrium level emerge in about 20% of the runs, for this setting, there are still about 10% of runs which lead to coordination on prices close to the monopoly level. However, supra-competitive prices almost completely disappear for five firms. In most runs with $k = 5$, the profit gain is close to zero, meaning that prices close to the Nash equilibrium level emerge.[7]

Close to Nash equilibrium outcomes for five firms do not depend on our particular choices of the learning rate and the exploration parameter. In Figure 4, we show heat maps for the mean (panel (a)) and standard deviation (panel (b)) of long-run profit gains across the batch runs with $k = 5$ firms, where the learning rate α and the exploration parameter β vary within a wide range of values. The figure shows that for all parameter constellations, the average profit gain is below $\Delta = 0.2$ and in large parts of the parameter space is well below $\Delta = 0.1$. Furthermore, the variation across runs is relatively small, with the standard deviation of profit gains consistently below 0.2.

In the Online Appendix B, we demonstrate that our finding is also in-

[7] Closer inspection of the distribution of Δ values shows that in less than 10% of the runs a Δ value above 0.25 emerges.



(a) (b)

Figure 4: Panel (a) shows (color-coded) mean values and panel (b) standard deviations over 100 batch runs of long-run values of profit gain Δ for $k = 5$ and a variation of α in the range $[0.0005, 0.01]$ with stepsize 0.0005 and β in $[5 * 10^{-6}, 10^{-4}]$ with stepsize $5 * 10^{-6}$.

dependent of our choice of using online gradient descent rather than experience replay for updating the weights of the neural networks in the DQN algorithm. In particular, considering analogous figures to Figures 3 and 4 for DQNs with experience replay (parameterized according to standard values from the machine learning literature), we show that also for this version of the DQN algorithm, the average profit gain decreases strongly as the number of firms goes up. The average gain is consistently low if five firms are in the market. If firms use DQNs with experience replay, already for markets with four competitors in the majority of runs, the profit gains are below $\Delta = 0.3$, such that the effect of an increase in k on average profit gains seems to be stronger for the case of DQNs with experience replay rather than online gradient descent.

To connect our findings with the existing literature, it should be noted that they differ qualitatively from results obtained in Waltman and Kaymak (2008), who show that in the framework of Cournot oligopolies tabular Q-learning yields quantities substantially below the level of the Nash equilibria for any number of firms between two and six. Apart from the facts that they used quantity rather than price competition and tabular Q-learning rather than DQNs, the main difference between their setup and ours is that the state firms take into account is always two-dimensional, consisting of the own quantity as well as the total quantity of competitors in the previous period. Since they use a model with linear inverse demand, where the distribution of quantities across competitors has no impact on the price, this

assumption seems reasonable. However, in our framework, with price setting and a non-linear demand curve, it seems essential that firms consider the individual prices of all competitors, as it is implemented in our simulations. Our observation that the profit gains of firms using DQNs decrease as the number of competitors go up is consistent with findings obtained in Hettich (2021) for a quite different DQN specification and parametrization as well as a very short time horizon for the training. Relating our finding to the experimental literature, it is interesting to note that a standard finding in experiments with human subjects is that, as observed in our simulations, the frequency of the occurrence of supra-competitive prices strongly decreases as the number of firms increases from two to four or five (e.g., Huck et al., 2004; Orzen, 2008).

4 Conclusion and policy implication

To the extent to which algorithms set supra-competitive prices, consumers are hurt. While it is well documented that firms use pricing algorithms, which algorithms are used is a black-box (c.f. Spann et al., 2024). To evaluate whether and how algorithmic pricing affects market outcomes, algorithms that are likely to be used have to be analyzed. We explore the properties of a deep Q-network pricing algorithm in an oligopoly setting. This algorithm is a state-of-the-art machine learning tool capable of dealing with high-dimensional state spaces that arise as more firms interact. In line with most recent simulation studies, we find that supra-competitive prices emerge in duopoly. However, prices quickly move toward (Nash) equilibrium levels as the number of firms increases.

Our findings suggest that competition authorities must be concerned about the emergence of supra-competitive prices due to the use of ML-based algorithms only if the market is highly concentrated with a very low number of relevant market participants. Hence, fostering market entry is an efficient strategy for keeping the undesirable effects of algorithms on price levels in check. However, avoiding supra-competitive prices in markets with two or three competitors is exceptionally challenging. Results reported in Calvano et al. (2020) or Asker et al. (2024) suggest that in such narrow markets, the price levels emerging from the interaction of algorithms of tabular Q-learning type depend strongly on the details of the algorithms. Exploring this issue further in the context of DQN is left for future research.

References

Abada, I. and X. Lambin (2023): "Artificial intelligence: Can seemingly collusive outcomes be avoided?" *Management Science*, 69, 5042–5065.

Asker, J., C. Fershtman, and A. Pakes (2022): "Artificial intelligence, algorithm design, and pricing," *AEA Papers and Proceedings*, 112, 452–56.

——— (2024): "The impact of artificial intelligence design on pricing," *Journal of Economics & Management Strategy*, 33, 276–304.

Barfuss, W. and J. M. Meylahn (2023): "Intrinsic fluctuations of reinforcement learning promote cooperation," *Scientific Reports*, 13, 1309.

Bengio, Y. (2012): "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade*, ed. by G. Montavon, G. Orr, and K.-R. Müller, Springer, chap. 19, 437–478.

Calvano, E., G. Calzolari, V. Denicoló, and S. Pastorello (2020): "Artificial intelligence, algorithmic pricing, and collusion," *American Economic Review*, 110, 3267–97.

——— (2021): "Algorithmic collusion with imperfect monitoring," *International Journal of Industrial Organization*, 79, 102712.

Competition and Market Authority (2018): "Pricing algorithms: Economic working paper on the use of algorihms to facilicate collusion and personalized pricing," CMA: Competition and Market Authority, CMA94.

Epivent, A. and X. Lambin (2024): "On algorithmic collusion and reward–punishment schemes," *Economics Letters*, 237, 111661.

European Commission (2017): *Report from the Commission to the Council and the European Parliament, Final report on the E-commerce Sector Inquiry*, Brussels: European Commission COM(2017) 229 final.

Goodfellow, I., Y. Bengio, and A. Courville (2016): *Deep learning*, Camgridge, Massachusetts: MIT press.

Hettich, M. (2021): "Algorithmic Collusion: Insights from Deep Learning," *SSRN Electronic Working Papers*.

Huck, S., H.-T. Normann, and J. Oechssler (2004): "Two are few and four are many: number effects in experimental oligopolies," *Journal of Economic Behavior & Organization*, 53, 435–446.

Klein, T. (2021): "Autonomous algorithmic collusion: Q-learning under sequential pricing," *The RAND Journal of Economics*, 52, 538–558.

Liu, X. (2023): "Dynamic coupon targeting using batch deep reinforcement learning: An application to livestream shopping," *Marketing Science*, 42, 637–658.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. (2015): "Human-level control through deep reinforcement learning," *Nature*, 518, 529–533.

Monopolies Commission (2018): "Biennial Report XXII: Digital change requires legal adjustments regarding price algorithms, the media sector and the supply of medicines," German Monopolies Commission.

Orzen, H. (2008): "Counterintuitive number effects in experimental oligopolies," *Experimental Economics*, 11, 390–401.

Qiu, D., Y. Ye, D. Papadaskalopoulos, and G. Strbac (2020): "A deep reinforcement learning method for pricing electric vehicles with discrete charging levels," *IEEE Transactions on Industry Applications*, 56, 5901–5912.

Spann, M., M. Bertini, O. Koenigsberg, R. Zeithammer, D. Aparicio, Y. Chen, F. Fantini, G. Z. Jin, V. Morwitz, P. P. Leszczyc, et al. (2024): "Algorithmic Pricing: Implications for Consumers, Managers, and Regulators," National Bureau of Economic Research, NBER 32540.

Waltman, L. and U. Kaymak (2008): "Q-learning agents in a Cournot oligopoly model," *Journal of Economic Dynamics and Control*, 32, 3275–3293.

Zhao, Z. and C. K. M. Lee (2022): "Dynamic pricing for EV charging stations: A deep reinforcement learning approach," *IEEE Transactions on Transportation Electrification*, 8, 2456–2468.

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)– Project-ID 317210226 – SFB 1283. The authors gratefully acknowledge the computing time provided to them on the high-performance computer Noctua 2 at the Paderborn Center for Parallel Computing (PC2). This is funded by the German Federal Ministry of Education and Research and the state governments participating on the basis of the resolutions of the Joint Science Conference (GWK) for the national high-performance computing at universities (www.nhr-verein.de/unsere-partner).

Online Appendix

A Details of the DQN Algorithm

A.1 Description of the algorithm

The function $Q_i(s_i, p; \theta_i)$ for firm i, determining the Q-values for action $p \in P$ in state s_i, is represented by a deep neural network with weight vector θ_i. This network is used to determine the greedy prices, i.e., the action $p_{i,t}$, which in a state $s_{i,t}$ is associated with the largest Q-value, and weights that are updated every period using stochastic gradient descent. We denote by $m = |P|$ the number of prices the firm can choose from. More precisely, we use a neural network with one input layer with k nodes, h hidden layers, each with $v_l,\ l = 1, ..h$ nodes, and a layer with m output nodes. The activation functions in the nodes of the hidden layer are sigmoid of the form $\phi(x) = \frac{1}{1+e^{-x}}$, whereas, as is common in the literature (see, e.g. Mnih et al., 2015), the activation function of the output nodes is the identity. Denoting by $\tilde{\theta}^l_{j_l, j_{l+1}}$ for $l = 0, .., h$ the weight between node j_l in layer l and j_{l+1} in layer $l+1$ (with $l = 0$ denoting the input layer and $l = h+1$ the output layer) and writing all weights as one vector of the form $\theta_i = \tilde{\theta}^l_{j_l, j_{l+1}}$ with $i = \left(\sum_{\tilde{l}=0}^{l-1} v_l \ v_{l+1}\right) + (j_l - 1)v_{l+1} + j_{l+1}$ the function represented by the neural network can be written as

$$Q_i(s_i, p; \theta_i) = \sum_{j_h=1}^{v_h} \tilde{\theta}^h_{j_h, j_{h+1}} \cdot \phi \left(\sum_{j_{h-1}=1}^{v_{h-1}} \dots \phi \left(\sum_{j_1=1}^{v_1} \tilde{\theta}^1_{j_1, j_2} \cdot \phi \left(\sum_{j_0=1}^{v_0} \tilde{\theta}^0_{j_0, j_1} s_{i, j_0} \right) \right) \right),$$

where $j_{h+1} \in \{1, .., m\}$ is the index of p in the set P of potential prices. The reward of firm i in period t is given by its scaled profit, i.e., $r_{i,t} = \chi \pi_{i,t}$, where we allow for scaling of the objective with factor $\chi > 0$ to improve learning behavior. Defining the loss function at t as

$$L_{i,t}(\theta_{i,t}) = \left(Q_i(s_{i,t}, p_{i,t}; \theta_i) - \left(r_{i,t} + \gamma \max_{p'} Q(s_{i,t+1}, p'; \tilde{\theta}_{i,t}) \right) \right)^2$$

the gradient is given by $q_{i,t} = \frac{\partial L_{i,t}(\theta_{i,t})}{\partial \theta_{i,t}}$. Here, $\tilde{\theta}_{i,t}$ denotes the weights of the target neural network, which is used for generating the Q-learning targets used in the loss function. Every τ^T period, the weights of the target network are set equal to the weights $\theta_{i,t}$ and then kept constant for the following τ^T periods.

Algorithm 1 Firm behavior

procedure DETERMINESTATE
  $s_{i,t} = (p_{1,t-1}, .., p_{k,t-1})$
procedure SETPRICE
  if $rnd < \epsilon_t$ **then** $p_{i,t} \leftarrow$ *random choice*
  else $p_{i,t} = \max_p Q(s_{i,t}, p; \theta_{i,t})$
procedure REWARD
  $r_{i,t} = \chi(p_{i,t} - c)q_{i,t}$
procedure UPDATEQ
  Add experience $(s_{i,t}, p_{i,t}, \pi_{i,t}, s_{i,t+1})$ *to* $\mathcal{D}_{i,t-1}$
  Drop $(s_{i,t-\mathcal{T}}, p_{i,t-\mathcal{T}}, \pi_{i,t-\mathcal{T}}, s_{i,t-\mathcal{T}+1})$ *from* $\mathcal{D}_{i,t-1}$
  $\rightarrow$ *backward window* $\mathcal{D}_{i,t}$
  Sample minibatch $\mathcal{D}_t^m$ *of size* n^m *from* $\mathcal{D}_t$
  For each $\{(s_{i,l}, p_{i,l}, \pi_{i,l}, s_{i,l+1})\} \in \mathcal{D}_{i,t}^m$
  Set $y_{i,l} = r_{i,l} + \gamma \max_{p'} \tilde{Q}(s_{i,l+1}, p'; \tilde{\theta}_{i,t})$
  and $L_{i,l} = (Q(s_{i,l}, p_{i,l}; \theta_{i,t}) - y_{i,l})^2$
  Average gradient: $\bar{g}_{i,t} = 1/n^m \sum_{l \in \mathcal{D}_t^m} \frac{\partial L_{i,l}}{\partial \theta_{i,l}}$
  $\theta_{i,t+1} \leftarrow \theta_{i,t+1} + \alpha \bar{g}_{i,t}$
procedure UPDATE$\tilde{Q}$
  if $t \mod \tau^T = 0$ **then**
    $\tilde{\theta}_{i,t+1} \leftarrow \theta_{i,t+1}$
procedure UPDATEGREEDYPARAMETER
  $\epsilon_{t+1} \leftarrow \epsilon_t e^{-\beta}$

Under online gradient descent, the weights are updated every period according to

$$\theta_{i,t+1} = \theta_{i,t} + \alpha g_{i,t},$$

where $\alpha > 0$ denotes the learning rate.

Under experience replay a minibatch of size n^m is drawn from a backward window $\mathcal{D}_{i,t}$ consisting of the last $\mathcal{T}$ observations:

$$\mathcal{D}_{i,t} = \{(s_{i,t-\mathcal{T}}, p_{i,t-\mathcal{T}}, \pi_{i,t-\mathcal{T}}, s_{i,t-\mathcal{T}+1}), ..., (s_{i,t}, a_{i,t}, \pi_{i,t}, s_{i,t+1})\}.$$

For each element in the minibatch, the corresponding gradient is calculated (using weights $\theta_{i,t}, \tilde{\theta}_{i,t}$), and the $\theta_{i,t}$ is updated using the average gradient of the minibatch.

To minimize the effect of heterogeneities of initial neural network weights θ_i on the variance of results across runs, we generated one default initial-

k	p^N	π^N	p^M	π^M
2	1.47	0.22	1.92	0.34
3	1.35	0.12	2.0	0.25
4	1.33	0.08	2.05	0.2
5	1.31	0.06	2.1	0.17

Table 1: Values of the prices and profits under Nash equilibrium and monopoly pricing for $k = 2, .., 5$

ization of weights, which is used for all runs, both with online gradient descent and experience replay. We verified that our results hardly change if stochastic initialization of weights is used instead.

The simulation setup is summarized in Algorithm 1 .

A.2 Parametrization

For the parameters of the market model, we follow Calvano et al. (2020) and set them to $a_i = 2, c_i = 1$ for $i = 1, .., k$, $a_0 = 0$ and $\mu = 0.25$. In Table 1, we provide the values of the prices and profits under Nash equilibrium and monopoly pricing for $k = 2, .., 5$.

The discretized price space P is given by an equidistant grid of $m = 18$ prices between $\underline{p} = p^N - \xi(p^M - p^N)$ and $\bar{p} = p^M + \xi(p^M - p^N)$ with $\xi = 0.6$.[8] The discount rate γ is set equal to 0.95. Hyper-parameters of the neural network are set as follows; see also Bengio (2012): We use neural networks with $h = 2$ hidden layers with the number of nodes in the four layers given by $v = (k, \lfloor\frac{2(k+m)}{3}\rfloor, \lfloor\frac{3(k+m)}{2}\rfloor, m)$. In the baseline specification, the learning rate is $\alpha = 0.001$, and the exploration parameter is $\beta = 6 \cdot 10^{-5}$. Under online gradient descent, the frequency of network update is $\tau^T = 1$. Under experience replay, the size of the minibatch is $n^m = 32$ drawn from a backward window of size $\mathcal{T} = 10^5$. The frequency of network update under experience replay is $\tau^T = 10^4$. Rewards are scaled by a factor of $\chi = 10$.

B Results for DQN learning with experience replay

In Figure 5, we show the box plots of the profit gain Δ for DQNs with experience replay and $k = 2, 3, 4, 5$ firms. Comparable to an algorithm with

[8] This value of ξ is larger than $\xi = 0.1$ chosen in Calvano et al. (2020) in order to avoid potential boundary effects that might occur during the DQN training.

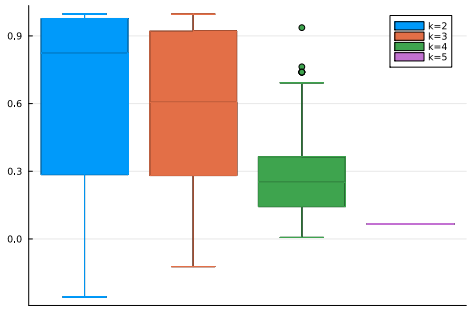
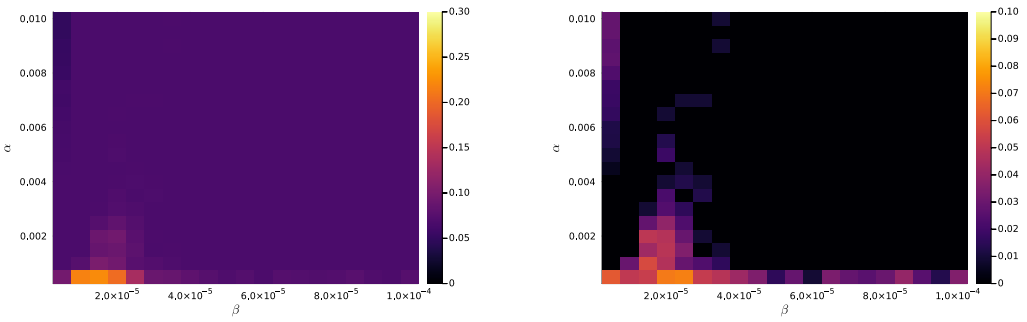


Figure 5: Boxplots of distribution of long-run values of profit gain Δ for DQNs with experience replay and $k = 2, 3, 4, 5$ firms and $\alpha = 0.001, \beta = 6 \cdot 10^{-5}$.

online gradient descent, profit gains fall as the number of firms increases. Profit gains are almost zero for five firms, confirming our main result. A further robustness test is shown in Figure 6, varying, again, the key parameters α and β for the case of five firms. For a large set of parameters, the mean values of the profit gain are close to zero, see Panel (a). The variance over the 100 batch runs is very small (c.f. Panel (b)). Considering the prices in these runs shows that in almost all runs the firms converge to the price in P that is closest to the Nash equilibrium price.



(a) (b)

Figure 6: Panel (a) shows (color-coded) mean values and panel (b) standard deviations over 100 batch runs of long-run values of profit gain Δ for DQNs with experience replay, $k = 5$ and a variation of α in the range $[0.0005, 0.01]$ with stepsize 0.0005 and β in $[5 * 10^{-6}, 10^{-4}]$ with stepsize $5 * 10^{-6}$.