Multi-agent Q-learning and regression trees for automated pricing decisions

Manu Sridharan   MSRIDHAR@MIT.EDU

Dept. of Electrical Engineering and Computer Science, MIT 38-401, Cambridge MA 02139 USA

Gerald Tesauro   TESAURO@WATSON.IBM.COM

IBM Thomas J. Watson Research Center, 30 Saw Mill River Rd., Hawthorne, NY 10532 USA

Abstract

We study the use of single-agent and multi-agent Q-learning to learn seller pricing strategies in three different two-seller models of agent economies, using a simple regression tree approximation scheme to represent the Q-functions. Our results are highly encouraging – regression trees match the training times and policy performance of lookup table Q-learning, while offering significant advantages in storage size and amount of training data required, and better expected scaling to large numbers of agents. Clear advantages are seen over neural networks, which yield inferior policies and require much longer training times. Our work is among the first to demonstrate success in combining Q-learning with regression trees. Also, with regression trees, Q-learning appears much more feasible as a practical approach to learning strategies in large multi-agent economies.

1. Introduction

Reinforcement learning in multi-agent environments is a challenging forefront of machine learning research that could have immense practical benefit in many real-world problems. Many application domains are envisioned in which teams of software agents cooperate to achieve a global objective. We also foresee significant applications for self-interested agents, for example, in electronic marketplaces, where economic software agents can interact with humans and/or other agents to maximize their own individual profits.

This paper investigates the use of Reinforcement Learning in multi-agent economies. Specifically, we study the use of Q-learning (Watkins, 1989) to learn pricing strategies in a competitive marketplace. Q-learning is an algorithm for learning to estimate the long-term expected reward for a given state-action pair. It has the nice properties that it does not need a model of the environment, and it can be used for on-line learning. With a lookup table representing the Q-function, the learning procedure is guaranteed to converge to the optimal value function and optimal policy.

The are two main challenges in using Q-learning in multi-agent systems. First, most of the existing convergence proofs only apply to single-agent, stationary Markov Decision Problems. (Important first steps in analyzing the two-agent case have been reported in (Littman, 1994) and (Hu and Wellman, 1998).) However, when multiple agents simultaneously adapt, each agent provides an effectively non-stationary environment for the other agents. Hence it is unknown in general whether any global convergence will be obtained in this case, and if so, whether such solutions are optimal. Second, we expect that large multi-agent systems will have states spaces that are too large for lookup tables to be feasible, and hence some sort of function approximation scheme seems necessary to represent the Q-function.

The present work combines single-agent and multi-agent Q-learning with tree-based function approximation in a model two-seller economy. The sellers alternately take turns setting prices, taking into account the other seller's current price. After the price has been set, the consumers choose either seller 1's product or seller 2's product, based on the current price pair. This leads to an instantaneous reward or profit given to the sellers. We assume that both sellers have full knowledge of the expected consumer response for any price pair, and moreover have full knowledge of both reward functions.

Q-learning is one of a variety of ways of endowing agents with "foresight," i.e. an ability to antic-

ipate long-term consequences of actions. Foresight was found in previous work (Tesauro and Kephart, 1999a,b) to improve profitability, and to damp out or eliminate the pathological behavior of unending cyclic "price wars," in which long episodes of repeated undercutting amongst the sellers alternate with large jumps in price. Such price wars were found to be rampant in prior studies of agent economy models (Kephart, Hanson and Sairamesh, 1998; Sairamesh and Kephart, 1998) when agents use "myopically optimal" or "myoptimal" pricing algorithms that optimize immediate reward, but do not anticipate any longer-term consequences. Q-learning in particular is a principled way to obtain deep lookahead, since the Q-function represents the cumulative discounted reward looking infinitely far ahead in time. In contrast, the prior work of (Tesauro and Kephart, 1999a) was based on shallow finite lookahead.

Q-learning with lookup tables was previously studied in (Tesauro and Kephart, 1999b). A single Q-learner playing against a myoptimal opponent always converged to an optimal policy. In the more interesting case of two agents simultaneously Q-learning against each other, convergence was again obtained in all three models, at least for small-to-moderate values of the discount parameter. In (Tesauro, 1999), preliminary studies of Q-learning with neural networks found reasonably good policies but excessively long training times. This is a potentially major drawback in an online scenario where agents must learn quickly to maintain profitability. The current study of regression trees as an alternate function approximator was motivated by the goal of improving on the training time of neural networks while generating policies of at least equal quality to those given by a neural network.

The remainder of this paper is organized as follows. Section 2 describes the structure and dynamics of the model two-seller economy, and presents three economically-based seller profit functions which are prone to price wars when agents myopically optimize their short-term payoffs. Section 3 describes the implementation of Q-learning in these model economies, and summarizes previous results of lookup table and neural network Q-learning. In Section 4, the algorithms used for constructing regression trees and adapting them to Q-learning in our study are presented. Section 5 presents the results of using Q-learning with regression trees and compares these results to those of section 3. Finally, section 6 summarizes the main conclusions and discusses promising directions and challenges for future work.

2. Model agent economies

Our models make a number of simplifying assumptions relative to the likely complexities of real agent economies. The economy is restricted to two sellers, competing on the basis of price, who offer similar or identical products to a large population of consumer agents. Prices are discretized and lie between a minimum and maximum price; there are typically ~ 100 possible prices. This renders the state space small enough to use lookup tables to represent the agents' pricing policies and expected profits. Time is also discretized; at each time step, the consumers compare the current prices of the sellers, and instantaneously and deterministically choose to buy from at most one seller. Hence at each time step, for each possible pair of prices, there is a deterministic profit obtained by each seller.

We also assume that the sellers alternately take turns adjusting their prices, rather than simultaneously setting prices. Alternating-turn dynamics is motivated by two considerations: (a) It ensures that there will be a deterministic optimal policy (Littman, 1994), and hence normal Q-learning, which yields deterministic policies, can apply. (b) In a realistic many-seller economy, it seems reasonable to assume that the sellers will adjust their prices at different times rather than at the same time (although probably not in a well-defined order).

The three economic models studied here are described in detail elsewhere [1]. In the first model, called the "Price-Quality" model (Sairamesh and Kephart, 1998), the sellers' products have different values of a scalar "quality" parameter, with higher-quality products being perceived as more valuable. At each time step, the consumers buy the lowest-priced product subject to constraints of a maximum allowable price and a minimum allowable quality. The substitutability of seller products enables direct price competition, and the "vertical" differentiation of differing quality values leads to asymmetries in the sellers' profit functions. Such asymmetries can result in unending cyclic price wars when the sellers employ myoptimal pricing strategies.

The second model, described in (Kephart, Hanson and Sairamesh, 1998), is an "Information-Filtering" model in which the two sellers offer news articles in partly overlapping categories. This model contains a "horizontal" differentiation of article categories. To the

---

[1] Descriptions and prior studies of these economic models are available on the Web at: www.research.ibm.com/infoecon/researchpapers.html.

extent that the categories overlap, there can be direct price competition, and to the extent that they differ, there are asymmetries that again lead to the potential for cyclic price wars.

The third model is the "Shopbot" model described in (Greenwald and Kephart, 1999), which models the situation on the Internet in which some consumers use a shopbot to compare prices of all sellers offering a given product, and select the lowest-priced seller. In this model, the sellers' products are identical, and their profit functions are symmetric. Myoptimal pricing leads the sellers to undercut each other until the minimum price point is reached. At that point, a new price war cycle can be launched, due to asymmetric buyer behavior, rather than seller asymmetries. Some buyers choose a random seller rather than bargain-hunt with the shopbot; this makes it profitable to abandon the low-price competition, and instead maximally exploit the random buyers by charging the maximum possible price.
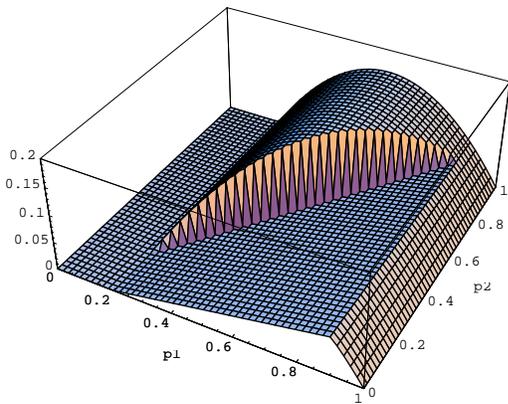
Figure 1. Sample profit landscape for seller 1 in Price-Quality model, as a function of seller 1 price p_1 and seller 2 price p_2.

An example seller profit function, taken from the Price-Quality model, is plotted in figure 1. This shows the instantaneous profit for seller 1, $U_1(p_1, p_2)$. The quality parameters are $q_1 = 1.0$, $q_2 = 0.9$ (i.e. seller 1 is the higher-quality seller). The myoptimal policy for seller 1, $p_1^*(p_2)$, is obtained for each value of p_2 by sweeping across all values of p_1 and choosing the value with the highest profit. For small p_2, the peak profit is obtained at $p_1 = 0.9$, whereas for larger p_2, there is eventually a discontinuous shift to the other peak, which follows the parabolic-shaped ridge along the diagonal.

The Information-Filtering and Shopbot models also

have similar profit landscapes. In all three models, it is the existence of multiple, disconnected peaks in the landscapes, with varying relative heights depending on the other seller's price, that leads to price wars when the sellers behave myopically.

In these models it is assumed for simplicity that the players have essentially perfect information. They can model the consumer behavior perfectly, and they also have perfect knowledge of each other's costs and profit functions. Hence the model is in essence a two-player perfect-information deterministic game, similar to games like chess. The main differences are that the payoffs are not strictly zero-sum, there are no terminating nodes in the state space, and payoffs are given to the players at every time step.

3. Single and Multi-agent Q-learning

3.1 Learning algorithm

The standard procedure for Q-learning is as follows. Let $Q(s, a)$ represent the discounted long-term expected reward (with discount parameter γ) to an agent for taking action a in state s. (The value of a reward expected at n time steps in the future is discounted by γ^n.) Assume that $Q(s, a)$ is represented by a lookup table containing a value for every possible state-action pair, and that the table entries are initialized to arbitrary values. Then the procedure for solving for $Q(s, a)$ is to infinitely repeat the following two-step loop:

1. Select a particular state s and a particular action a, observe the immediate reward r for this state-action pair, and the resulting state s'.

2. Adjust $Q(s, a)$ according to the following equation:

$$\Delta Q(s, a) = \alpha[r + \gamma \max_b Q(s', b) - Q(s, a)] \quad (1)$$

where α is the learning rate parameter. A variety of methods may be used to select state-action pairs in step 1, provided that every state-action pair is visited sufficiently often. When α is decreased over time with an appropriate schedule, the above procedure is guaranteed to converge to the correct values for stationary MDPs.

In our economic models, the distinction between states and actions is somewhat blurred. It will be assumed that the "state" for each seller is sufficiently described by the other seller's last price, and that the "action" is the current price decision. This should be a sufficient state description because no other history is needed either for the determination of immediate reward, or

for the calculation of the myoptimal price by the fixed-strategy player. We have also redefined r and s' for the two-agent case as follows: let s' be the state that is obtained, starting from s, of one action by the Q-learner and a response by the opponent. Likewise, r is defined as the sum of the two rewards obtained after those two actions. These modifications were introduced so that the state s' would have the same player to move as state s. (A possible alternative to this, which has not been investigated, is to include the side-to-move as additional information in the state-space description.)

3.2 Results of lookup table learning

Detailed results of lookup table-based Q-learning are presented in (Tesauro and Kephart, 1999b). In brief, single-agent Q-learning in all three models was found to always yield exact convergence to a stationary optimal solution (as expected). The resulting Q-derived policies always outperformed a myopic strategy when tested against myopic opposition, and the expected profit increased montonically with γ. In many cases, Q-learning had the side benefit of also improving the myopic opponent's expected profit. This improvement is due to the Q-learner learning to abandon undercutting behavior more readily as the price decreases. The price-war regime is thus smaller and confined to higher average prices, leading to a closer approximation to collusive behavior, with greater expected profits for both sellers.

For simultaneous Q-learning by both sellers, the procedure utilized was to alternately adjust a random entry in seller 1's Q-function, followed by a random entry in seller 2's Q-function, using the same formalism presented above. As the Q-functions evolved, the policies were correspondingly updated so that they optimized the agents' current Q-function. In modeling the two-step payoff r to a seller in equation 1, the opponent's current policy was used, as implied by its current Q-function.

Simultaneous Q-learning in the Price-Quality model yielded robust convergence to a unique pair of policies, independent of γ, identical to the solution found by shallow lookahead in (Tesauro and Kephart, 1999a). In the Shopbot model, exact convergence of the Q-functions was only found for $\gamma < 0.7$. For $\gamma \geq 0.7$, there was very good approximate convergence, in which the Q-functions converged to stationary solutions to within small random fluctuations. Different solutions were obtained at each value of γ. For small γ, a symmetric solution is generally obtained (in which the shapes of $p_1(p_2)$ and $p_2(p_1)$ are identical), whereas a broken symmetry solution, similar to the

Price-Quality solution, is obtained at large γ. There was a range of γ values, between 0.1 and 0.2, where either a symmetric or asymmetric solution could be obtained, depending on initial conditions. The asymmetric solution seems counter-intuitive because one would expect that symmetric profit functions would lead to symmetric policies. Finally, in the Information-Filtering model, simultaneous Q-learning produced exact or good approximate convergence for $0 \leq \gamma \leq 0.5$. For larger γ, no convergence was obtained. The Q-derived policies yielded reduced-amplitude price wars, and montonically increasing profitability for both sellers as a function of γ, up to $\gamma = 0.5$.
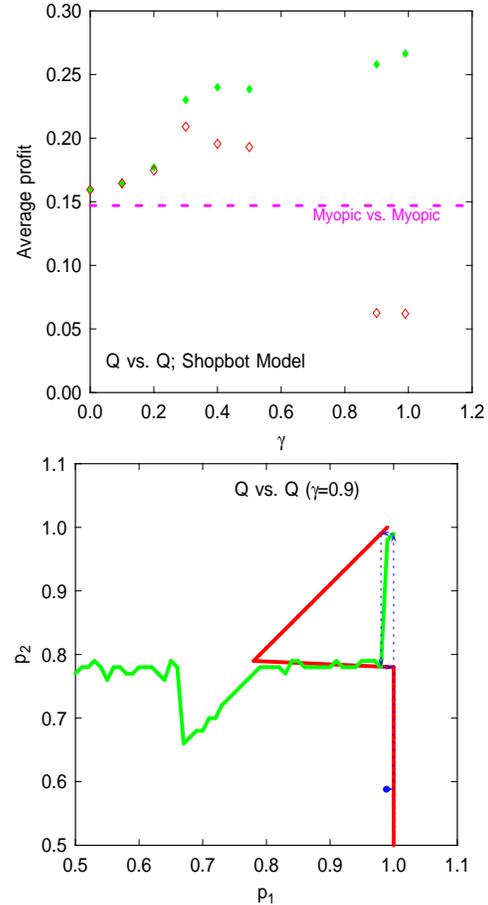
Figure 2. Results of simultaneous Q-learning with lookup tables in the Shopbot model. (a) Average utility per time step for seller 1 (solid diamonds) and seller 2 (open diamonds) vs. discount parameter γ. Dashed line indicates baseline myopic vs. myopic expected utility. (b) Cross-plot of Q-derived price curves at $\gamma = 0.9$. Dashed line and arrows indicate a sample price dynamics trajectory.

Figure 2 is illustrative of the results of simultaneous Q-learning. The left figure plots the average profit for both sellers in the Shopbot model. The right figure plots Q-derived price curves (at $\gamma = 0.9$) of seller 1

and seller 2 against each other. The system dynamics can be obtained by alternately applying the two pricing policies. This can be done by an iterative graphical construction, in which for any given starting point, one first holds p_2 constant and moves horizontally to the $p_1(p_2)$ curve, and then one holds p_1 constant and moves vertically to the $p_2(p_1)$ curve. For these particular curves, the graphical construction leads to a very short cyclic price war, indicated by the dashed line. The price-war behavior begins at the price pair $(1, 1)$, lasts only a couple of steps, and then drops to $p_2 \sim 0.8$. At this point seller 1 resets its price to $p_1 = 1.0$ and the cycle repeats. The price war amplitude is diminished compared to myopic vs. myopic play, where a long price war would persist all the way to a minimum price point ~ 0.58.

3.3 Difficulties of neural network training

Some preliminary results of combining Q-learning with neural networks were reported in (Tesauro, 1999). The neural nets typically appeared to reach peak profitability in a few hundred sweeps through the training cases (corresponding to a few hours of CPU time). The policies were reasonably good at this point, and qualitatively similar to the lookup table policies, but the quality of approximation of the Q-function was poor, as indicated by large Bellman error. With much further training (out to several days of CPU time), the Bellman error improved significantly, but there was no improvement in policy profitability. It is possible that, with enough additional training, further improvements in profitability might be found, but it appeared that the required training times would be prohibitively long.

4. Q-learning with regression trees

Our regression tree algorithm employs the simplest conceivable heuristics: we use axis-parallel splits, select splits that minimize variance, and approximate the function by constant values in the leaf nodes. A brief description appears below; more details can be found in (Breiman et al., 1984).

The trees are constructed in a "batch" mode using a fixed set of training cases. Each training case has d input attribute values, and an associated function value which may be adjusted during training. Given a training set N, a regression tree is constructed recursively as follows: First, the average a and the variance v of the function values of the cases are calculated. If $|N|$ is less than a given threshold or if v is sufficiently small, the algorithm terminates, returning a leaf node which approximates the function by the constant a. Other-

wise, the best axis-parallel split of the cases is found by examining all possible splits on each attribute. The best split is defined as follows: consider a split that divides N into sets N_1 and N_2, with respective variances v_1 and v_2. The split which minimizes the quantity $|N_1|v_1 + |N_2|v_2$ is the best split. A new internal node defined by this split is then created. The training set is separated into two subsets, and two subtrees for the new node are created recursively. Finally, the algorithm returns the node and terminates.

We performed some initial investigations of minimal error-complexity pruning, as described in (Breiman et al., 1984). However, it was found that setting a minimum of five cases in each leaf node and not using any pruning gave better results. Minimal error-complexity pruning seems ill-suited to our problem, perhaps because the policies generated by using the tree are more important than the accuracy of the tree's approximation of the function. Other pruning algorithms have not been investigated.

In our problem, the training cases consist of random price pairs (corresponding to uniform random exploration of the state-action space), plus an additional "knowledge-engineered" binary attribute which is set to 1 when the Q-learner's price is less than the opponent's price, and 0 otherwise. This helps the tree represent the critically important "undercutting" discontinuity present in all three models.

The function values for the cases are initialized to the seller's two-step immediate reward for that state and action (as in Section 3). An initial tree is built from these cases. Then, repeated sweeps are performed through the set of training cases, in which the case values Q_i are adjusted according to:

$$\Delta Q_i = \alpha[r + \gamma \max_b Q(s', b) - Q_i] \qquad (2)$$

where the max-Q value for the successor state is found using the current tree. A new tree is built after each sweep through the training set. The algorithm terminates after a fixed number of sweeps. Training runs typically used a fixed learning rate α, which seemed to give good results even though convergence theorems require decreasing α with time.

In the case of multiple Q-learners, the tree updates are performed as follows. First, the sweep through the training set of each Q-learner is done, with the most recent policy of the other Q-learners (determined from their most recently constructed tree) used to calculate the immediate reward r. Then, the trees of all the Q-learners are reconstructed. This method promotes

consistency of the Q-functions, as all Q-learners have access to equally recent information about other Q-learners.

One slightly different algorithm was also examined, which involved building a tree as above, and then further refining the leaf node values. A random new point in the space is chosen, and the corresponding leaf node is determined. Let a be the current leaf node estimate of the function, k be the number of cases that fell in that leaf node during previous training, and f be the function value of the new point. The leaf node's estimate is then updated according to: $a' = (a * k + f)/(k + 1)$. This process continues for a number of new points, after which a new tree is built from a newly constructed training set. This algorithm worked well in the case of a single Q-learner, but for multiple Q-learners was not as good at producing stable policies as the batch updating algorithm described above.

5. Results

In the case of single-agent Q-learning, the batch training algorithm converged rapidly, within a few dozen iterations, taking ~ 1 minute or less of CPU time on a fast RS/6000 workstation. This represents a huge improvement over neural network training, which ranged from a few hours to several days of CPU time. Furthermore, in all three economic models, regression tree learning was consistently able to match the exact optimal policies that were obtained from lookup-table Q-learning. (Again, this is superior to neural nets, which usually were not able to find the optimal policies.)

These optimal policies were found with training set sizes that were relatively small fractions of the total state space size. Typically we find that data sets $\sim$ 20-25% of the state space size in the Price-Quality and Information-Filtering models, and $\sim$ 40-50% in the Shopbot model, are sufficient to generate exact or near-optimal policies. As shown in figure 3, the algorithm's behavior as a function of training set size is quite good. The policy profitability quickly asymptotes at the level of the lookup table policy, and for smaller training sets, there is graceful degradation: in most cases a near-optimal policy is found, however, there is an increasing chance that a poor solution will be obtained as the training set size is reduced.

We can also see in figure 3 that the trees grow to reasonable sizes (much smaller than the corresponding lookup tables) as a function of training set size. For example, in the Shopbot model, trees with approximately 300 nodes (or 150 leaf nodes) were able to generate optimal policies. Trees with approximately 600 nodes generated optimal policies in the Price-Quality model, with the higher quality seller as the Q-learner.
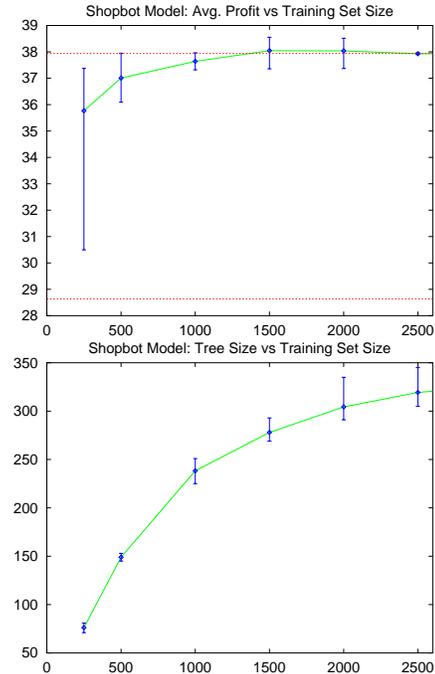
Figure 3. Results of single-agent Q-learning with regression trees in the Shopbot model, $\gamma = 0.5$. (a) Average Q-learner profit per time step as function of training set size. Error bars represent min and max profit over 10 runs with different randomly-generated training sets. Lower dashed line represents average profit in myopic vs. myopic play. Upper dashed line indicates average profit of exact optimal policy obtained by lookup table Q-learning. (b) Average tree size as a function of training set size.

Most of our results were obtained with the minimum number of training cases per leaf node ("Minobjs") set to 5. This lenient criterion resulted in a large number of leaf nodes, with relatively good constant-value approximation within each leaf node. We have also investigated more stringent stopping criteria by increasing Minobjs. This results in smaller trees, with potentially cruder function approximation within the leaf nodes. Generally we find robust behavior with increasing Minobjs, as illustrated in figure 4. Increasing Minobjs from 5 to 20 was found to reduce the tree size by nearly a factor of 3, with only a slight decrease in policy profitability.

An illustration of the quality of regression tree function approximation compared to an exact lookup table solution is shown in figure 5. The tree correctly fits the diagonal discontinuity along the undercutting line $p_Q < p_{MY}$ (aided greatly by the "knowledge-
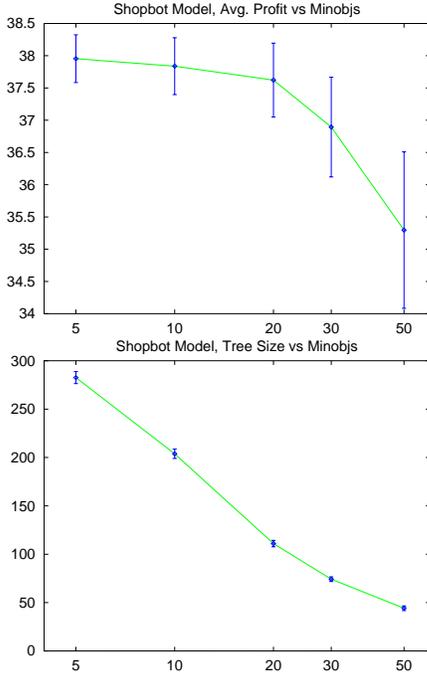
Figure 4. Results of varying Minobjs, the minimum no. of cases per leaf node. Single-agent Q-learning with regression trees in the Shopbot model, $\gamma = 0.5$, 1500 training cases. (a) Average Q-learner profit per time step as function of Minobjs. Error bars represent min and max profit over 10 runs with different randomly-generated training sets. (b) Average tree size as a function of Minobjs.

engineered" input attribute), and unlike neural networks, it also correctly fits the second discontinuity in the small p_Q regime. This is important in obtaining the optimal policy. Neural nets fit the flat and smoothly curving portions of the landscape better, but this is irrelevant to finding the optimal policy.

The results of multi-agent Q-learning using regression trees were also promising. In the Price-Quality model, results were identical to those generated with lookup tables: robust convergence was found to a self-consistent pair of policies independent of γ. In the Information-Filtering model, convergence was obtained for γ up to 0.5, with cumulative profits for both sellers increasing with higher γ, exactly as in the lookup table case. Results for the Shopbot model were mixed. For γ up to 0.2, convergence to symmetric policies was obtained. Using the leaf-node updating algorithm described briefly in Section 4, the same asymmetric solutions were obtained for $\gamma > 0.2$. Using the batch training algorithm with large γ, the policies of the sellers remain symmetric for some time, after which they suddenly become asymmetric and no convergence is obtained. Although these results are not

completely understood, they seem to correspond well with the lookup table results, where it was also seen that the symmetric solution for multi-agent Q-learning was highly unstable.
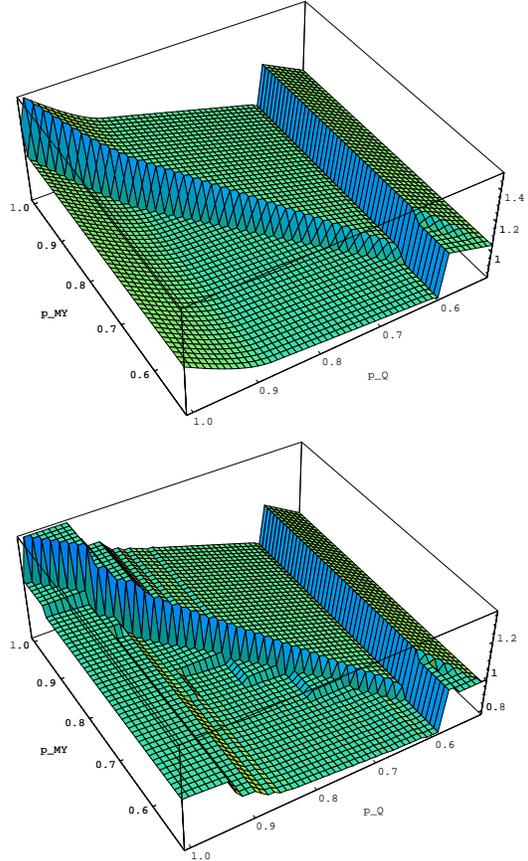
Figure 5. Plot of the Q-function for single-agent Q-learning vs. a myopic opponent in the Shopbot model at $\gamma = 0.7$. p_Q is the Q-learner's price and p_MY is the myopic player's price. (a) Exact optimal Q-function obtained by lookup table Q-learning. (b) Regression tree approximation to the Q-function. The tree was trained on 1500 cases and contained at least 10 cases per leaf node.

Training times in the multi-agent case increased, but convergence was still usually obtained in under 5 minutes. Partly this was due to having to use a smaller value of $\alpha \sim 0.1$, whereas in the single agent case, it was often possible to go as high as $\alpha \sim 1$ and still obtain convergence. Tree sizes increased slightly in multi-agent Q-learning tests, which is reasonable since the profit landscapes are more complicated. In the Shopbot model, trees that generated optimal policies ranged from 350-400 nodes, and trees with approximately 700 nodes generated optimal policies for the higher-quality seller in the Price-Quality model.

6. Conclusions

We have studied a simple combination of tree-based function approximation with single-agent and multi-agent Q-learning, and have found highly encouraging results in three different models of agent economies, each with a different mechanism for generating price wars. This suggests that the success of the algorithm is not due to specific details of any single model, and it is likely to be more generally applicable in agent economies.

The main contributions of this work are twofold. First, an important open question in Reinforcement Learning research is how to combine RL methods such as Q-learning with nonlinear function approximation. Most of the empirical research on this topic has involved neural networks. Our work, along with (Wang and Dieterich, 1999), is among the first to demonstrate success in combining Q-learning with regression trees. Our results show clear advantages over the neural net approach, and thus provide impetus for further studies of tree-based methods.

The second contribution is that with regression trees, Q-learning appears much more feasible as a practical approach to learning strategies in multi-agent economies. With lookup tables, Q-learning generally performed well in terms of training time and policy profitability, both in the single-agent and multi-agent cases, but is expected to scale poorly with the number of agents. Regression trees are expected to offer much better scaling. We find that they train quickly, achieve profits equal or very close to the lookup table policies, and offer significant advantages in terms of tree size and amount of training data required.

One clearly important direction of future research is to examine how well regression trees perform in economic models with more agents and more realistic details. (Preliminary results indicate that in a three-seller case, regression trees still seem to perform well with a manageable increase in the size of the trees.) It also seems likely that more sophisticated methods for generating splits and leaf-node function approximation, such as oblique splits and linear function approximation, could be of clear benefit. It will also be important to resolve how to adapt our batch training methodology to online learning, which is inherently incremental.

Finally, we have observed empirical behavior of multi-agent Q-learning, such as approximate but not exact convergence, that has no analog in ordinary Q-learning, where the Bellman equation is known to be a contraction mapping, leading to a unique global attractor with zero residual error. The empirical results here, together with those of (Sandholm and Crites, 1995) for Prisoner's Dilemma, suggest that more interesting theoretical results may be available in the multi-agent case. Depending on the payoff functions and on γ, there may be a single global attractor, or multiple basins of attraction, or no attractor dynamics at all. Also, the attractors may be small finite regions or limit cycles rather than points, with non-zero asymptotic residual error.

References

L. Breiman et al., *Classification and Regression Trees*. Monterey CA: Wadsworth, 1984.

A. Greenwald and J. O. Kephart, "Shopbots and pricebots." Proceedings of IJCAI-99, 506-511, 1999.

J. Hu and M. P. Wellman, "Multiagent reinforcement learning: theoretical framework and an algorithm." Proceedings of ICML-98, 1998.

J. O. Kephart, J. E. Hanson and J. Sairamesh, "Price-war dynamics in a free-market economy of software agents." Proceedings of ALIFE-VI, Los Angeles, 1998.

M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," Proceedings of the Eleventh International Conference on Machine Learning, 157–163, Morgan Kaufmann, 1994.

J. Sairamesh and J. O. Kephart, "Dynamics of price and quality differentiation in information and computational markets." Proceedings of the First International Conference on Information and Computation Economics (ICE-98), 28–36, ACM Press, 1998.

T. W. Sandholm and R. H. Crites, "On multiagent Q-Learning in a semi-competitive domain." Proceedings of IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems, Montreal, Canada, 71–77, 1995.

G. J. Tesauro and J. O. Kephart, "Foresight-based pricing algorithms in agent economies." *Decision Support Sciences*, to appear, 1999(a).

G. Tesauro and J. O. Kephart, "Pricing in agent economies using multi-agent Q-learning." Proceedings of: Workshop on Decision Theoretic and Game Theoretic Agents, London, England, 5-6 July 1999(b).

G. Tesauro, "Pricing in agent economies using neural networks and multi-agent Q-learning." Proceedings of: IJCAI-99 Workshop on Learning About, From and With Other Agents, Stockholm, Sweden, 2 Aug. 1999.

X. Wang and T. G. Dieterich, "Efficient value function approximation using regression trees." Proceedings of: IJCAI-99 Workshop on Statistical Machine Learning for Large-Scale Optimization, Stockholm, Sweden, 31 Jul. 1999.

C. J. C. H. Watkins, "Learning from delayed rewards." Ph. D. thesis, Cambridge University, 1989.