Abstract

A shopbot is a software agent whose goal is to maximize buyer's satisfaction through gathering automatically the price and quality information of products as well as the services from on-line sellers. For the sellers on the Internet, there is a software agent, called a pricebot, as a counterpart of a shopbot. This paper adopts Q-learning, one of the model-free reinforcement learning methods, as a price-setting algorithm for pricebots. A Q-learned agent increases profitability and reduces the cyclic price wars when compared with the agents using the myoptimal (myopically optimal) pricing strategy. In the process of Q-learning, an agent needs to select a sequence of state-action pairs for learning to converge. When the uniform random method in selecting state-action pairs is adopted for a pricebot, the number of accesses to the Q-tables to obtain the convergence of the optimal Q-value is quite large. Therefore, the uniform random selection (URS) is not appropriate for universal on-line Q-learning in a realistic environment, since URS reflects the uncertainty of exploitation for the optimal policy and doesn't consider a compromise between exploration and exploitation in reinforcement learning.

In this paper, we propose Mixed Nonstationary Policy (MNP), which consists of both the auxiliary Markov process and the original Markov process. A Q-learning agent to which MNP is applied uses a stationary greedy policy to control the original Markov process which is intended to exploit certainly the estimate of the optimal actions evaluated by both the auxiliary Markov process and the original Markov process in order to solve the problem of the uncertainty of exploitation for the optimal policy. Our experimental results show that MNP converges to the optimal values about 2.6 times faster than URS on the average. We show that the pricebots with MNP can be extended to universal on-line Q-learning agents by achieving a compromise between exploration and exploitation.

Keywords: Reinforcement learning, Q-learning, Multi-agent systems, Agent economies

1. Introduction

As more and more information on consumer products and services is available on the Internet, buyers have no choice but to rely on software agents that are capable of providing convenience and profitability to them by obtaining useful and proper information quickly and automatically. A shopbot is a web agent that queries multiple on-line sellers to gather automatically the price and other attributes of products as well as the services from on-line sellers. Recently shopbots have expanded over a wide range of products. For the sellers on the Internet, there is a software agent called a pricebot, as a counterpart of a shopbot. A pricebot uses automatic pricing strategies to be able to bring the sellers with profitability and satisfaction [1]. In order to make a decision on profitable prices, a pricebot needs to employ a dynamic price setting algorithm.

The agent economy described in this paper is based on the Internet where it is difficult for agents to know one another well while the environment undergoes changes during learning. Thus a model-free learning method for on-line training is essential to this economic environment. This paper adopts Q-learning [2] for pricebots to learn the optimal pricing policy against other agents. Q-learning is one of the model-free reinforcement learning methods which address the question of how an autonomous agent that senses and acts in its environment can learn to choose optimal actions to achieve its goals. In the Markov decision process, a Q-learning agent can directly learn its optimal policy by interacting with the environment without knowing any reward function or any state transition function. A Q-learned agent is capable of anticipating a long-term consequence of actions. As a result, it is able to gain...
more profits and to reduce the pathological behavior of unending cyclic price wars which are found when the “myoptically optimal” (or myoptimal) pricing strategy that doesn’t have deep look-ahead is embedded into agents [3].

One of the challenges that arise in reinforcement learning is the trade off between exploration and exploitation. An agent has to exploit what it already knows in order to obtain a reward, but it also has to explore in order to make better action selections in the future. But it is not possible to explore and to exploit, at the same time, with any single action selection, hence a conflict could occur between exploration and exploitation [4].

In the process of Q-learning, a single agent has to select a sequence of state-action pairs to decide the optimal policy. When the agent adopts the uniform random selection (URS) of state-action pairs amongst all possible table entries, the uncertainty problem of exploitation for the optimal policy may occur. Hence URS is not a technical solution for a compromise between exploration and exploitation in reinforcement learning. As a result, URS requires a great deal of accesses to Q-tables for learning and hence URS is not appropriate for universal on-line Q-learning in a realistic environment even though Q-learning can be used for on-line training. In order to solve the uncertainty problem, we propose a new method called Mixed Nonstationary Policy (MNP) for a compromise between exploration and exploitation in reinforcement learning. When an agent adopts MNP for Q-learning to converge, it uses a stationary greedy policy to control the original Markov process which is intended to exploit certainly the estimate of the optimal actions evaluated by both the auxiliary Markov process and the original Markov process. As a result, a compromise between exploration and exploitation in reinforcement learning can be achieved and the convergence to the optimal values can be obtained fast with a quite smaller number of accesses to Q-tables. Our experimental results show that MNP converges to the optimal values about 2.6 times faster than URS on the average, provided that a fixed number of steps L in MNP is adjusted properly to have a balance between exploration and exploitation in reinforcement learning.

The remainder of this paper is organized as follows. Section 2 explains the agent economy consisting of shopbots and pricebots, and presents some results of price dynamics for two agents when both of them use the myoptimal pricing algorithm. Section 3 describes some issues in designing a single agent Q-learning algorithm and shows an implementation of a single Q-learned agent against a myoptimal agent. Section 4 illustrates MNP in detail and Section 5 presents and analyses the experimental results for Q-learned agents with MNP and URS. Finally, Section 6 summarizes the conclusions and some future work.

2. Shopbot and Pricebot Agent Economy

2.1. Model

The agent economy which this paper adopts is composed of shopbot and pricebot agents on the Internet where consumers use shopbots to gather prices and other attributes from all online vendors, and where sellers use pricebots to maximize their profits employing dynamic price setting algorithms. We assume that there is a single homogeneous product that is offered for sale by S sellers and there are B buyers, with B ≫ S. Each seller s potentially resets its price ps with rate ρs at random times and each buyer b orders for purchase at random times. The value of the product to buyer b is vb and the cost of production for seller s is cs. ps is the price offered by seller s. Buyer b’s utility for the product can be written as follows:

\[ u_s(p_s) = \begin{cases} v_b - p_s & \text{if } p_s \leq v_b \\ 0 & \text{otherwise} \end{cases} \] (1.0)

We assume that buyers are not willing to maximize their utilities. Instead we assume that buyers consider the prices offered by on-line sellers using either the ‘Any Seller’ strategy or the ‘Bargain Hunter’ strategy [5]. The Buyer population consists of a mixture of buyers with a fraction w1 using the Any Seller strategy and a fraction w2 using the Bargain Hunter strategy, where w1 + w2 = 1.

A seller s’s expected profit per unit time, \( \pi_s \), is a function of the price vector \( p \) as follows:

\[ \pi_s(p) = (p_s - c_s)D_s(p) \] (1.1)

where \( D_s(p) \) is the rate of demand for the product produced by seller s and thus it can be defined as follows:

\[ D_s(p) = \rho B h_s(p)g(p) \] (1.2)

where \( \rho B \) is the overall buyer rate of demand \( (p = \sum \rho_b) \), \( h(p) \) is the probability that seller s is selected by buyers and \( g(p) \) denotes the fraction of buyers whose valuations satisfy \( v_b \leq p_s \). If \( \rho B = 1 \), without loss of generality, the expected profit of seller s can be described as follows:

\[ \pi_s(p) = (p_s - c_s)h_s(p)g(p) \] (1.3)

Because \( h_s(p) \) depends on the buyer distribution \( (w_1, w_2) \), it can be rewritten as follows:

\[ h_s(p) = w_1 f_1^s(p) + w_2 f_2^s(p) \] (1.4)

where \( f_1^s(p) \) denotes the probability that buyers using the Any Seller strategy select seller s and equals to \( 1/S \) because it is independent of the ordering of sellers’ prices. But \( f_1^s(p) \) is...
the probability that buyers using the Bargain Hunter strategy select seller \( s \). Hence \( f_s^*(p) \) depends on the relative ordering of the sellers’ prices. Therefore, we can define \( f_s^*(p) \) as:

\[
f_s^*(p) = \frac{1}{\tau_s(p) + 1} \delta_{\lambda_s(p), 0} \quad (1.5)
\]

where \( \tau(p) \) is the number of sellers charging the same price as \( s \), excluding \( s \) itself; \( \delta \) is a Kronecker delta function and it is equal to 1 whenever \( i = j \), and 0 otherwise; \( \lambda_s(p) \) is the number of sellers charging a lower price than \( s \). If all buyer valuations are the same \( (v_b = v) \), we can describe \( g(p) = \Theta(v - p) \) as follows:

\[
\Theta(v - p) = \begin{cases} 
1 & \text{if } p \leq v \\
0 & \text{otherwise}
\end{cases} \quad (1.6)
\]

The preceding equations can be used to define \( \pi_s(p) \) in terms of the distribution of strategies and valuations within the buyer population as follows. We assume that \( v_b = v \) and \( c_s = c \) for all buyers \( b \) and all sellers \( s \), respectively.

\[
\pi_s(p) = \begin{cases} 
(p_i - c) h_s(p) & \text{if } p_s \leq v \\
0 & \text{otherwise}
\end{cases} \quad (1.7)
\]

where

\[
h_s(p) = \frac{w_i}{S} + \frac{1}{\tau_s(p) + 1} \delta_{\lambda_s(p), 0} \quad (1.8)
\]

The equations described so far can be found in [1, 5]. The agent economy which we consider is restricted to two competing sellers. Prices are discrete and lie between a minimum price \( p_{\min} \) and a maximum price \( p_{\max} \) for our simulations. We also assume that sellers alternately take turns to adjust their prices.

2.2. Myoptimal Pricebots

The myoptimal pricing strategy performs an exhaustive search for the optimal price that maximizes its expected profit \( \pi_s \) [6]. Figure 1 illustrates the cyclic price wars when both of two pricebots use the same myoptimal pricing strategy. These myoptimal pricebots undercut each other by price quantum \( \epsilon \) until the end point 0.58 is reached. At the end point, a new price war cycle is ignited by resetting its price to the buyer valuation \( v = 1.0 \). Because the myoptimal pricing strategy is based on the zero look-ahead, a pathological behavior of unending cyclic price wars occurs as in Figure 1. In Figure 1 we did not draw all the points made by both agents. Otherwise, we hardly see the nature of counter offers.

3. Single Agent Q-learning

3.1. Design of Single Pricebot Q-learning Algorithm

The activities of pricebots in the agent model economy explained in the previous section can be regarded as the Markov decision process.

**Definition** A Markov Decision Process is a tuple \( <S, A, r, p> \), where \( S \) is the discrete state space, \( A \) is the discrete action space, \( r : S \times A \rightarrow R \) is a reward function of an agent, where \( R \) is the set of real number, and \( p : S \times A \rightarrow \Delta \) is a transition function, where \( \Delta \) is the set of probability distributions over state space \( S \) [7].

We assume that the system in this paper is a deterministic Markov decision process. The task of an agent is to learn a policy, \( \pi : S \rightarrow A \), for selecting its next action based on the current observed state. In a Markov decision process, an agent senses the current state and chooses a policy, \( \pi \) to maximize \( V(s, \pi) \) which is the discounted cumulative reward achieved by policy \( \pi \) from initial state \( s \). Such a policy is the optimal policy and it is denoted by \( \pi^* \):

\[
\pi^* = \arg \max_{\pi} V(s, \pi), \quad (\forall s) \quad (2.0)
\]

In real agent economies, it is difficult for agents to know the perfect knowledge on a reward function and a state transition function. Thus an agent needs to learn directly about its optimal policy through interacting with the environment. Reinforcement learning is a powerful and practical solution for these problems. Q-learning is one of the reinforcement learning methods which do not need a model of the environment and can be used for on-line learning. To explain the basic idea of Q-learning, we can define the function, \( Q^*(s, a) \) as follows:
\[ Q'(s, a) = r(s, a) + \gamma \sum_{s'} V(s', \pi^*) \]  

(2.1)

where \( \gamma \in [0,1) \) is a discount factor. By this definition, \( Q^*(s, a) \) is the total discounted cumulative reward that can be achieved starting from state \( s \) and applying action \( a \) as the first action and then following the optimal policy thereafter. Then using equation (2.1) we can write \( V(s, \pi^*) \) as follows:

\[ V(s, \pi^*) = \max_a Q'(s, a) \]  

(2.2)

Therefore, if \( Q^*(s, a) \) is known, then the optimal policy can be found. There is a proof about the convergence of Q-learning for deterministic Markov decision processes in [8].

We define some important issues in designing simulation of single agent Q-learning in this paper.

- The initial values of the elements in a Q-table are set to the immediate reward values.
- An action is an agent’s pricing.
- A state is \((a_1', a_2')\), where \(a_1'\) is an action by Q-learner and \(a_2'\) is a response action by the other agent.
- An immediate reward is the sum of the two rewards obtained after those two actions.

To learn about Q-values, a Q-learning agent needs to maintain a Q-table for each of \( m \) states, where \( m \) is the total number of states. For a Q-learned agent, a Q-table has rows corresponding to \( a_1' \in A_1' \), columns corresponding to \( a_2' \in A_2' \) where \( A_1' \) and \( A_2' \) are the sets of agent 1’s actions and agent 2’s actions, respectively. The total number of entries a Q-learned agent needs to learn is \( m \times |A_1'| \times |A_2'| \), where \(|A_1'|\) and \(|A_2'|\) are the sizes of \( A_1' \) and \( A_2' \), respectively.

3.2. Q-learned Pricebot vs. Myoptimal Pricebot

When a Q-learned agent competes against a myopic opponent, it abandons the price war more quickly as the prices decrease. Therefore, the amplitude of the price war gets smaller and a higher average profit is guaranteed [9]. An illustrative example of the results of competition between a Q-learned pricebot and a Myoptimal pricebot is shown in Figure 2 and Table 1. As in Figure 1, we omit some points from the figure. The Q-learned agent is trained with a discount parameter \( \gamma = 0.5 \). In Figure 2, the price war begins at the price pair \((1.0, 1.0)\) and continues until Q-learner’s pricing value reaches at 0.78. When the myoptimal agent sets its price to 0.775, the Q-learned agent abandons the price war by setting its price to the minimum end point 0.58 and resets its price to 1.0. These results show that the amplitude of the price war can be smaller than that of the price war when both agents use the myopic strategy. Table 1 depicts the average profits of the simulations in Figures 1 and 2. We can see that the average profit of the Q-learned agent is higher than all other agents.

![Figure 2. Price dynamics for a Q-learned pricebot and a myoptimal pricebot](image)

Table 1. Average profits in two pricebots simulations

<table>
<thead>
<tr>
<th>Pricebots</th>
<th>Average profits</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Myopic, Myopic)</td>
<td>(0.248, 0.248)</td>
</tr>
<tr>
<td>(Q-learned, Myopic)</td>
<td>(0.329, 0.322)</td>
</tr>
</tbody>
</table>

4. Extension to Universal On-line Q-learning

One of the challenges that arise in reinforcement learning is how to handle a compromise between exploration and exploitation. There are detailed exploration and exploitation: Exploration ensures that all admissible state-action pairs are explored often enough to satisfy the Q-learning convergence theorem, and exploitation seeks to minimize the cost-to-go function by following a greedy policy [10]. Q-learning needs a strategy to provide a compromise between these two conflicting objectives.

To decide the optimal policy in the process of Q-learning, a single agent has to select a sequence of state-action pairs. It may use URS to choose state-action pairs amongst all possible table entries [3, 8, 11]. But URS for convergence of Q-learning reflects the uncertainty of exploitation for the optimal policy and doesn’t consider a compromise between exploration and exploitation in reinforcement learning. URS is hardly a technical solution for reinforcement learning. As a result, the number of accesses to Q-tables for the convergence to the optimal values is increased greatly. Therefore, URS is inappropriate for universal on-line Q-learning in a realistic environment even though Q-learning is the learning method which can be used for on-line training.

In this paper, we propose MNP in [12], in order to solve the uncertainty of exploitation for the optimal policy and to extend to universal on-line Q-learning through decreasing the number
of accesses to Q-tables for the convergence of the optimal values. In MNP, the auxiliary Markov process and the original Markov process are performed alternately. The stationary greedy policy determined by Q-learning controls the original Markov process. MNP starts in a state of the auxiliary process and chooses actions by following it. Then the original controlled process takes over, and it goes back and forth alternately as shown in Figure 3. We assume that the transition probability between a pair of states of the auxiliary Markov process and the original Markov process is not considered because Q-learning is implemented in a deterministic environment in this paper. The auxiliary Markov process is based on URS to choose a sequence of state-action pairs during Q-learning. It operates for a fixed number of steps \( L \). The time spent during the operation of the original controlled process is increased by \( L \) at every switch.

![Figure 3. Process transitions in MNP](image)

When a Q-learning agent adopts MNP for learning to converge, it uses the stationary greedy policy to control the original Markov process. Because each of the estimates of the optimal actions is guaranteed to be exploited for certain during the original Markov process, the uncertainty of exploitation for optimal policy can be solved. As a result, a compromise between exploration and exploitation in reinforcement learning can be achieved and the convergence of the optimal values can be obtained faster with a smaller number of accesses to the Q-tables, provided that MNP is tuned with a proper fixed number of steps \( L \) to achieve a compromise between exploration and exploitation in reinforcement learning. Thus Q-learning with MNP is appropriate for universal on-line training.

### 5. Experimental Results

In this section, we compare a Q-learning pricebot with MNP with a Q-learning pricebot with URS. We used the java language for experiments. MNP was implemented with multi-threading. For our experiments the parameters are used as in Table 2. We tested various values of \( L \), the fixed number of steps to analyze the performance results of MNP.

Figure 4 shows the average trials over the Q-tables from table Q(1.000, 0.995) to table Q(0.775, 0.770). In Figure 4, the Q-learning agent using MNP when \( L = 0.1 \) converges to the optimal Q-values faster among others and also makes decision for the optimal policy about 2.6 times faster than that using URS. We can see that the agent using MNP with a small \( L \) value such as \( L = 0.002 \) needs a considerable amount of accesses to Q-tables for learning to converge. The reason is that the auxiliary Markov process cannot explore unknown states and actions sufficiently during such a small fixed number of steps, even though the original Markov process achieves the exploitation of states and actions that have already been learned. When \( L \) is greater than or equal to 1.6 in MNP, the agent tends to converge to the optimal Q-value with a similar number of trials for each state-action pair to those with URS. This happens because if \( L \) is large it gives sufficient time for the auxiliary Markov process to learn the optimal policy. Thus the performance of MNP depends on the value of \( L \).

#### Table 2. Parameters used for experiments

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
<th>Meanings</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v )</td>
<td>1.0</td>
<td>buyer's valuation of the product</td>
</tr>
<tr>
<td>( c )</td>
<td>0.5</td>
<td>seller’s production cost</td>
</tr>
<tr>
<td>( w_r )</td>
<td>0.25</td>
<td>the fraction of buyers using the Any Seller strategy</td>
</tr>
<tr>
<td>( w_y )</td>
<td>0.75</td>
<td>the fraction of buyers using the Bargain Hunter strategy</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>0.5</td>
<td>the discounted parameter</td>
</tr>
<tr>
<td>( \varepsilon )</td>
<td>0.005</td>
<td>the price discretization interval</td>
</tr>
<tr>
<td>( p_{mín} )</td>
<td>0.58</td>
<td>the minimum price which sellers can offer</td>
</tr>
</tbody>
</table>

![Figure 4. The average number of trials needed to obtain the optimal Q-values for all the Q-tables](image)
Figure 5 shows the average number of trials for each of the Q-tables when the agents use URS, MNP with $L=0.002$, MNP with $L=0.1$, and MNP with $L=1.6$. We have tested for a very wide range of values of $L$, and Figure 5 shows for some typical values of $L$. The agent using MNP with $L=0.002$ which incurs a lack of exploration in reinforcement learning needs a great deal of accesses to most of the Q-tables compared with the agent using URS, while the agent using MNP tuned with $L=1.6$ which causes the exploitation by the original Markov process to be a good-for-nothing brings to converge to the optimal Q-value with a similar number of trials for each state-action pair to the agent using URS. There is only one case that URS outperformed MNP with $L=0.1$; table Q(0.795, 0.790). However, for all other cases MNP with $L=0.1$ showed better performance.

6. Conclusions

When an agent called a pricebot which brings high profitability to on-line sellers adopts Q-learning as price setting algorithm in agent-mediated Electronic Commerce, it needs to select a sequence of state-action pairs for Q-learning to converge. When a Q-learning agent uses URS which reflects the uncertainty of exploration for the optimal policy and which doesn’t provide a compromise between exploration and exploitation in reinforcement learning, the number of accesses to Q-tables for the convergence of the optimal Q-values becomes quite large. Therefore, URS is inappropriate for universal on-line Q-learning in a realistic environment.

In this paper we proposed MNP for universal on-line Q-learning. The experimental results have shown that Q-learning using MNP converges to the optimal Q-values much faster than URS, provided that a fixed number of steps $L$ is adjusted properly to get a compromise between exploration and exploitation in reinforcement learning.

One of the weaknesses in Q-learning is that the space requirement is exponential in the number of agents. Thus for an economic environment consisting of a large number of agents, the research on efficient representation for the action space is necessary. The lookup table representations for large-scale problems in real agent economies should also be reexamined. To do so, the research on combining reinforcement learning methods with effective function approximators is required as future work.

References


