RetailNet: Enhancing RETAIL of Perishable Products with Multiple Selling Strategies via Pair-Wise Multi-Q Learning

Xiyao Ma 1 Fan Lu 1 Xiajun Pan 2 Yanlin Zhou 1 Xiaolin Li 1

Abstract

We propose RetailNet, an end-to-end reinforcement learning (RL)-based neural network, to achieve efficient selling strategies for perishable products in order to maximize retailers’ long-term profit. We design Pair-wise Multi-Q network for Q value estimation to model each state-action pair and to capture the interdependence between actions. Generalized Advantage Estimation (GAE) and Entropy are incorporated into the loss function for balancing the tradeoff between exploitation and exploration. Experiments show that RetailNet efficiently produces the near-optimal solution, providing practitioners valuable guidance on their inventory replenishment, pricing, and products display strategies in the retailing industry.

1. Introduction

Efficient selling strategies for perishable products, i.e., fresh fruits, dairy products, meat, etc., is crucial for retailer profitability. Perishable products lose value over time and become outdated at the end of their shelf lives. Empirical evidence shows that price discount on old products increases customer demand and profit (Goyal & Giri, 2001). However, price discount may incur the competition between fresh and old products and result in adverse financial performance if ignored. Thus, a significant challenge is to maximize the profit by considering the competition between co-existing products of different ages (Ferguson & Koenigsberg, 2007).

In addition to discounting, retailers can also manipulate display settings to promote profit. For instance, retailers may simply place old products in front of fresh ones on the shelves and promote the sales of old products. Consumers, who do not search their favorite products actively, tend to purchase the old products. Nevertheless, this display setting may lead to some freshness sensitive consumers leaving the store without purchasing any product. To choose the right display setting is vital for the retailer to improve profits.

How discounting the old products and choosing a display setting to improve the profitability depends on the inventory replenishment policy. The inventory replenishment problem can be represented as a Markov Decision Process (MDP). Prior work uses RL and Q-learning algorithms with value and policy iteration for MDP problems (Watkins, 1989; Sutton et al., 1998; Raju et al., 2003). However, they are only capable of handling small problems with limited state and action spaces. Deep Reinforcement Learning algorithms that learn by trial-and-error have been shown robust and capable of solving broad domain of issues with larger state and action spaces, such as Atari games (Mnih et al., 2013), board games (Silver et al., 2017), and even protein folding (Li et al., 2018). Nevertheless, it is still challenging when multiple actions involve as it is hard to capture and model the interdependence and relevance among the actions precisely (Wang & Yu, 2016)(He et al., 2015).

The co-existence of fresh and aged products complicates the replenishment decision with the additional dimension of choosing the display setting and setting price discount factor. This calls for novel algorithms to solve a real-world retailing problem demanding computational resources and storage.

We propose RetailNet and RetailNet++ with Pair-wise Multi-Q network to model each state-action pair and the interdependence among these actions. To further improve the exploration of the action, a fundamental challenge in RL accompanied by large-scale state and action spaces, we adapt our proposed model in Asynchronous Advantage Actor-Critic (A3C) algorithm with entropy regularization (Mnih et al., 2016).

Experimental studies for cases, where the DP can solve, show that the profit achieved by RetailNet/RetailNet++ ap-
proaches the optimal profit with much shorter time. The developed algorithm is efficient in solving real-world retailing systems. The main contributions are: (1) Motivated by a practical retailing problem, we develop a complete model of inventory replenishment and selling for perishable products by exploiting consumers purchasing behavior. We create a simulator for training and evaluation; (2) We propose pair-wise Multi-Q network to model each state-action pair together and capture the interdependence among the multiple actions for more accurate value estimation; (3) Extensive experiments demonstrate the accuracy and efficiency of our RetailNet/RetailNet++.

2. Problem Formulation

Similar to (Ferguson & Koenigsberg, 2007), we consider a retailer selling perishable products with a lifetime of two periods. The products deteriorate in freshness over time. At the beginning of each period, the retailer purchases the products at a unit cost of $C_f$ from a reliable supplier, which is delivered immediately (no lead time) and sells at price $p$. We refer to these products as fresh products with the freshness level of $Q_f$. At the end of each period, unsold fresh products are carried over to the next period at the unit cost of $C_b$ and the freshness level drops to $Q_d$ with the remaining lifetime of one period. These products are sold as old products at a discounted price of $(1 - \rho)p$. At the end of each period, unsold old products expire, and the retailer disposes them at a unit cost of $C_d$.

Consumers are heterogeneous and characterized by their willingness to pay ($\alpha$) for one unit of freshness level. We assume $\alpha$ is uniformly distributed over $[0, \pi]$ as in (Ferguson & Koenigsberg, 2007). A consumer with valuation $\alpha$ derives a utility of $U_F(\alpha) = \alpha Q_f - p$ from consuming a fresh product and $U_O(\alpha) = \alpha Q_d - p(1 - \rho)$ from consuming an old product. Without loss of generality, we assume that the utility of buying nothing is equal to zero. Every consumer chooses the product which maximizes her utility provided that it is non-negative. Based on the valuation on the freshness of the products, customers can be divided into four categories:

- **O**: Customers with utility $U_O(\alpha) > 0 > U_F(\alpha)$ only purchase old products.
- **F**: Customers with utility $U_F(\alpha) > 0 > U_O(\alpha)$ only purchase fresh products.
- **OF**: Customers with utility $U_O(\alpha) > U_F(\alpha) > 0$ prefer old products but would still purchase fresh products when old products are out of stock.
- **FO**: Customers with utility $U_F(\alpha) > U_O(\alpha) > 0$ prefer fresh products but would still purchase old products when fresh products are out of stock.

Let $P_O$, $P_F$, $P_{OF}$, and $P_{FO}$ denote the proportion of customers of types $O$, $F$, $OF$ and $FO$, respectively. Table 1 characterizes these values corresponding to the range of the price discount parameter $\rho$.

Based on consumers’ behavior, when both types of products are in stock, customers of types $F$ and $FO$ buy fresh products and customers of type $O$, and $OF$ buy old products. In contrast, customers of types $F$, $FO$, $OF$ purchase fresh products when old products are out of stock, and customers of types $O$, $FO$, $OF$ purchase old products when fresh products are out of stock.

In each period, $N$ customers visit the store and deplete the inventory. Depending on the product category, $N$ may be constant or stochastic following a distribution. Like (Honhon & Seshadri, 2013), we use the fluid model to calculate the number of consumers of each type: $NP_O$, $NP_F$, $NP_{OF}$, $NP_{FO}$. The retailer decides the optimal order quantity $y_t$ of fresh products at the beginning of each period $t$ to maximize the average profit in an infinite time horizon, as shown in Figure 1.

![Figure 1. Illustration of the Retailing System](image)

As consumers may be active or passive in searching products, the retailer can also influence the product choice of customers through product display, by manipulating the ease with which customers can grab fresh and old products from the store shelves. We consider the following five display settings observed in practice:

- **Setting A**: Fresh and old products are displayed in a common area such that the products are equally reachable for customers;
- **Setting B**: Old products are displayed in the front while fresh products are displayed in the back of shelves such that old products are easier to reach;
- **Setting B’**: Fresh products are displayed in the front while old products are displayed in the back of shelves such that fresh products are easier to reach;
Table 1. \( P_O, P_F, P_{OF}, P_{OF} \) calculations as a function \( F \) of the discount parameter \( \rho \), where \( F \) follows a uniform distribution.

<table>
<thead>
<tr>
<th>Discount Range</th>
<th>( P_O )</th>
<th>( P_F )</th>
<th>( P_{OF} )</th>
<th>( P_{FO} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Low</td>
<td>0</td>
<td>( 1 - F(\frac{\rho}{\rho_0}) )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Low</td>
<td>( 1 - \frac{\rho_0}{\rho} ) ( \frac{\rho_0}{\rho} )</td>
<td>( F(\frac{\rho_0}{\rho} \frac{\rho_0}{\rho}) - F(\frac{\rho}{\rho_0}) )</td>
<td>0</td>
<td>1 - ( F(\rho_0^2 \rho_0^2) )</td>
</tr>
<tr>
<td>Medium</td>
<td>( \frac{\rho_0}{\rho} ) ( \rho_0 )</td>
<td>( F(\frac{\rho}{\rho_0}) - F(\rho_0^2 \rho_0) )</td>
<td>0</td>
<td>( 1 - F(\rho_0^2 \rho_0) )</td>
</tr>
<tr>
<td>High</td>
<td>( \frac{\rho_0}{\rho} ) ( 1 )</td>
<td>( F(\rho_0^2 \rho_0) - F(\frac{\rho}{\rho_0}) )</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2. Consumer choice in all five product display settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>( F_0 )</th>
<th>( O_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>( P_F + P_{FO} )</td>
<td>( P_O + P_{OF} )</td>
</tr>
<tr>
<td>B</td>
<td>( 1 - \beta P_F + (1 - \beta)P_{FO} )</td>
<td>( P_O + P_{OF} + \beta P_{FO} )</td>
</tr>
<tr>
<td>B'</td>
<td>( P_F + P_{FO} + \beta P_{OF} )</td>
<td>( (1 - \beta)P_O + (1 - \beta)P_{OF} )</td>
</tr>
<tr>
<td>C</td>
<td>( P_F + P_{FO} + P_{OF} )</td>
<td>0</td>
</tr>
<tr>
<td>C'</td>
<td>( P_F + P_{FO} + P_{OF} )</td>
<td>0</td>
</tr>
</tbody>
</table>

The retailer’s one-period profit is given by:

\[
\Pi(y_t; I, N) = \rho_S F(y_t; I, N) + p(1 - \rho)S_O(y_t; I, N) - C_d(I - S_O(y_t; I, N)) - C_f y_t - C_h I
\]

\[
= \rho_S F(y_t; I, N) + (p(1 - \rho) + C_d)
+C_f y_t - (C_h + C_d)I
\]

where \( C_f, p, C_h, \rho, C_d \geq 0 \), \( S_F(y_t; I, N) \) and \( S_O(y_t; I, N) \) denote the sales of fresh products and old products of each time period, respectively, as function of the quantity of fresh products replenished \( y_t \), available inventory of old products \( I \) and the total quantity of customers \( N \). The detail of computing \( S_F \) and \( S_O \) is out of scope. However, we provide the link if readers are interested.

The retailer’s objective is to maximize the long-term average profit over a finite time horizon by optimally (1) ordering the fresh products \( y_t \), (2) choosing the display setting \( d_t \), and (3) selecting the discount value \( \rho_t \). The profit-maximization problem is as follows.

\[
\max_{y_t, d_t, \rho_t} \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} E_N \left[ \Pi(y_t) \right] \tag{2}
\]

Some retailers have a consistent display setting and discount value across periods, so the only decision is to replenish their inventory in each period. nd the maximization problem is as follows.

\[
\max_{y_t, d_t, \rho_t} \frac{1}{T} \sum_{t=1}^{T} E_N \left[ \Pi(y_t) \right] \tag{3}
\]

Even for this basic model, we cannot obtain the optimal solutions efficiently using the traditional dynamic method owing to the large-scale state and action spaces. Therefore, we propose RetailNet and RetailNet++ to respectively obtain the near optimal inventory replenishment policy and simultaneously optimize the inventory replenishment, display setting and discount value.

\( ^1 \)Available at: https://sites.google.com/view/retailnet0/appendix
3. Proposed RetailNet

The retailer’s profit-maximizing problem can be represented as a Markov Decision Process (MDP). Before presenting the novel models, we first discuss the fundamental components of the proposed RetailNet as follows.

**State**: We denote state $s_t$ as the quantity of old products $I_t$ at the beginning of period $t$ (carried over from the previous period).

**Action**: At the beginning of period $t$, observing the current state $s_t$ (as the input), RetailNet performs a single action $a_t = y_t$, i.e., the order quantity of fresh products, and RetailNet++ performs multi-actions $a_t = [y_t, \rho_t, d_t]$, i.e., the order quantity of fresh products, the discount value on old products, and the display setting.

**Reward**: The immediate reward is the current period profit denoted as $r_t$, which is essentially the one-period profit $\Pi(y_t; I_t, N_t)$ calculated using Eq. (1). The accumulated Reward $R_t$ up to period $t$ can be computed with a reward discount factor $\gamma$, given Bellman Equation(Bellman, 1957):

$$R_t = r_t + \gamma R_{t+1}$$

3.1. RetailNet

Some retailers do not want to change the product display setting and discount value across periods, so the inventory replenishment is the only decision they can make in each period.

In this scenario, taken the current state $s_t$ with parameters $\theta_a$ as input, policy network $\pi(a_t|s_t; \theta_a)$ in RetailNet generates a single selling strategy $a_t$ on the quantity of fresh products. Meanwhile, a value network $V(s_t; \theta_v)$ is leveraged to evaluate the current state $s_t$ with parameters $\theta_v$. As depicted in Fig. 2(a), we realize our policy and value networks in a *multi-task* setting. The lower layers are shared across these two networks; the top layers are task-specific with different layers. Replenishment policy outputs an action probability vector $a_t$, and value network outputs a scalar $v_t$ as a critic to evaluate the current state $s_t$ as follows:

$$h = W_h(s) + b_h,$$  \hspace{1cm} (5)

$$a = Softmax(W_a h + b_a),$$  \hspace{1cm} (6)

$$v = W_v h + b_v,$$  \hspace{1cm} (7)

where $W$ and $b$ are trainable weights and bias in the neural network, respectively.

3.2. RetailNet++

When retailers adjust the discount value and display setting periodically, the profit can be further improved. We propose RetailNet++ to generate multiple actions simultaneously including replenishment quantity $y_t$, price discount value $\rho_t$, and display setting $d_t$. It uses the same policy network but outputs multiple actions by using different dimensional weights and bias in higher layers, as shown in Fig. 2(b).

In the case of large state and action space containing different kinds of actions, it is challenging for Q network to model the state and actions with an accurate value estimation owing to the intertwined actions. To solve the action ranking problem, (He et al., 2015) proposed Deep Reinforcement Relevance Network (DRRN) with two separate networks to estimate value for state and actions, respectively. However, it ignores the specific correlation between each state-action pair.

To tackle the issue, we propose Pair-wise Multi-Q network to model each $(s_t, a_t)$ state-action pair together and capture the interdependence among the actions. As shown in Fig. 2(b), we use Multi-Q network $q_i = Q_i(s_t, a_t)$ with the same architecture but different weights to obtain multiple Q values for each state-action pair. In practice, each pair-wise Q-network is realized by two-layer fully-connected networks with Dropout (Srivastava et al., 2014) and LeakyReLU as the nonlinear function. The final Q value is a weighted sum of individual $Q_i$ computed as follows:

$$Q_{final}(s_t, a_t) = \sum_{i=1}^{M} \phi_i q_i = \Phi^T Q$$

where $\Phi = [\phi_1, \phi_2, ..., \phi_M] \in R^M$ is the interdependence-aware weight vector, $Q = [q_1, q_2, ..., q_M] \in R^M$ is the Q value vector by concatenating the outputs of Pair-wise Multi-Q network, and $M$ is the total number of actions.

Then we exploit a bi-directional GRU network to model $\Phi$:

$$\Phi = BiGRU(q_i, h_i)$$

where $h_i$ is the current hidden state. We further obtain the final weight $\Phi$ by taking the average pooling across the bi-direction. Through the pair-wise interaction function, i.e., inner product, we compute $Q_{final}$ as the final Q value estimation on state and actions, which will be further incorporated in policy gradient loss function that we discuss next.

3.3. Policy loss and value loss

**Policy Loss**: For policy gradient update, the value function decides the magnitude of every update step (Sutton et al., 2000). An accurate estimation of value function that reflects expected future return makes model converge steadily and faster to the global optimal, but it is difficult to estimate an action with low variance and bias. For example, using reward as the value estimation is the most direct way and can reflect the expected return with small bias. However, the gradient may have a huge variance because of the accumulated
reward over several time periods. An alternative approach is to employ an advantage function to lower the variance by subtracting a baseline, which can be approximated with a value network. Nevertheless, introducing additional neural networks may incur bias. To address the issue, inspired by (Schulman et al., 2015), we incorporate generalized advantage estimation (GAE) into our policy loss function to balance the bias and variance with multi-step estimation as follows:

\[
\hat{A}_t^{GAE}(\gamma, \lambda) = \delta_t + (\gamma \lambda) \delta_{t+1} + \ldots + (\gamma \lambda)^{T-t+1} \delta_{T-1} = \sum_{i=0}^{\infty} (\gamma \lambda)^i \delta_{t+i}^v
\]

\[
\delta_i^v = r_t + \gamma V(s_{t+1}; \theta_v) - V(s_t; \theta_v)
\]

where \(\lambda\) is a tradeoff factor between bias and variance. Furthermore, to avoid local minima and collapsing on a single choice of action, an entropy term is incorporated in policy loss function for encouraging exploration given(Mnih et al., 2016):

\[
H(\pi(a_{t}^{i}|s_{t}; \theta_a)) = -\log \pi(a_{t}^{i}|s_{t}; \theta_a) \pi(a_{t}^{i}|s_{t}; \theta_a)
\]

Therefore, we have a total policy loss function for updating the parameters \(\theta_a\) of the policy network:

\[
\mathcal{L}(\theta_a) = -\hat{A}_t^{GAE(\gamma, \lambda)} - \mu H(\pi(a_{t}^{i}|s_{t}; \theta_a))
\]

\[
= - \sum_{i=0}^{\infty} \log \pi(a_{t}^{i}|s_{t}; \theta_a) (\gamma \lambda)^i \delta_{t+i}^v
\]

\[
- \mu \log \pi(a_{t}^{i}|s_{t}; \theta_a) \pi(a_{t}^{i}|s_{t}; \theta_a)
\]

where \(\mu\) controls the magnitude of entropy for action \(a_{t}^{i}\).

**Value Loss:** We train value/Pair-wise Multi-Q network \(V(s; \theta_v)\) by updating the parameters \(\theta_v\) to minimize the mean square loss function over time periods:

\[
\mathcal{L}(\theta_v) = 0.5E(R_t - V(s; \theta_v))^2
\]

As shown in Algorithm 1, we train our proposed RetailNet/RetailNet++ with A3C algorithm in an asynchronous update manner. The global shared parameters \(\theta_a\) and \(\theta_v\) are updated with thread-specific gradient \(\theta_a'\) and \(\theta_v'\) when the thread episode terminates, and update thread-specific parameters \(\theta_a''\) and \(\theta_v''\) by coping global parameters \(\theta_a\) and \(\theta_v\). Note that we use \(V(s_t; \theta_v)\) for value estimation in RetailNet, while we use \(Q_{final}(s_t, a_t)\) for value estimation in RetailNet++.

4. Experiments and Results

For retailer stores selling daily use perishable products such as milk, bread, the number of visiting consumers is relatively stable; while other retailers face more dynamic consumers. In this section, we evaluate our model under deterministic and stochastic demand cases by interacting with a simulated environment of the retailing system, respectively.

Prior work on perishable products mainly focus on inventory replenishment policy assuming retailers use FIFO (first-in-first-out) or LIFO (last-in-first-out) fulfillment policy.

<table>
<thead>
<tr>
<th>Group 1</th>
<th>p</th>
<th>C_I</th>
<th>C_h</th>
<th>C_d</th>
<th>Q_I</th>
<th>Q_h</th>
<th>MaxN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.0</td>
<td>0.5</td>
<td>0.10</td>
<td>0.2</td>
<td>10.0</td>
<td>4.0</td>
<td>4.0</td>
</tr>
<tr>
<td>2</td>
<td>3.5</td>
<td>0.3</td>
<td>0.05</td>
<td>0.1</td>
<td>8.0</td>
<td>3.5</td>
<td>10.0</td>
</tr>
</tbody>
</table>

Table 3. Parameter settings across all experiments

Two Groups of Parameters shown in Table 3 are used across experiments. Values of parameters are selected based on
We use the Profit Gap as the performance measurement: $\text{Profit Gap} = \frac{\text{optProfit} - \text{Profit}}{\text{optProfit}} \times 100\%$. Where $\text{Profit}$ is the profit achieved by our proposed RetailNets, and $\text{optProfit}$ is the optimal profit computed analytically in deterministic demand case and obtained using DP method in stochastic demand case, respectively. We use a 32-core CPU to train our RL agents, with a learning rate of 0.0001. We will release our code after review session.

4.1. Experiment with Deterministic Demand

In the deterministic demand case, we have a fixed number of consumers visiting a retailer store $N$. We analytically prove that there exists an optimal policy that no old products are discarded in any period. Our proposed RetailNet outputs the order quantity of fresh products for a given display setting and discount value, and our proposed RetailNet++ outputs the display setting and discount value in addition to the order quantity of fresh products, which realizes a near-optimal profit as shown in Columns 1 of Table 4. Column 5 reports the time used for the RetailNets. Note that the DP method does not give the optimal solution ("NA" is noted correspondingly) after running one day for the multiple actions scenario, where RetailNet++ outputs a good solution in around 29 minutes.

<table>
<thead>
<tr>
<th>Models</th>
<th>Profit Gap (%)</th>
<th>Profit</th>
<th>optProfit</th>
<th>Time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RetailNet</td>
<td>0.0000</td>
<td>0.0001</td>
<td>0.0001</td>
<td>28.47</td>
</tr>
<tr>
<td>RetailNet++</td>
<td>0.0000</td>
<td>0.0001</td>
<td>0.0001</td>
<td>296.4</td>
</tr>
</tbody>
</table>

Table 4. Profit and time by RetailNet and RetailNet++ with parameter Group 1 and 2 in Table 3.

4.2. Experiment with Stochastic Demand

In this section, we conduct the numerical study for the case where the number of consumers arriving in a store follows uniform and beta distributions, respectively, and report the results in Tables 5 and 6. We compare our model with two intuitive baseline model for modeling the current state and actions as follows:

Q-Single: Given the state and three actions as the input, use a neural network to output an $Q$ value.

Q-Aver: Use three neural networks to model each state-action pair and average the $q$ values as the final $Q$ value.

Similar to the previous experiment results, RetailNets can produce a near-optimal solution more efficiently. Meanwhile, by modeling the state and three actions with Multi-Q network, our model is showed to make a higher profit than the two baselines and converge more stably and rapidly, as shown in Figure 3. Furthermore, the profits made by RetailNet++ are higher than the maximal optimal profit made by single selling strategies, showing that RetailNet++ can make more profit than RetailNet. Column 5 reports the time used by our RetailerNets/DP. "NA" means that the DP cannot produce a solution after running a day.

We also perform another experiment by comparing the results of RetailNet++ with two different $y_t$ precision, 0.1 and 0.0256.
solve the problem optimally. The most related work to ours mainly focus on heuristic policies because it is difficult to out (LIFO) manner; both fresh and old products are priced

time of our RetailNet++ only takes less than 10 minutes
to preciser actions used, the profit with precision of 0.05 is

Table 5. Profit and time by RetailNet and RetailNet++ with cus-
tomer demand $N \sim U[0, 4]$ and $N \sim U[0, 10]$ with parameter
settings of Group 1 and and Group 2 in Table 3, respectively.

Table 6. Profit and time by RetailNet and RetailNet++ with cus-
tomer demand following a beta distribution $\text{Beta}(\alpha, \beta)$, where $\alpha = 1.0$ and $\beta = 0.5$ with parameter settings of Group 1 and Group 2 in Table 3, respectively.

0.05 respectively. As shown in Table 5 and Table 6, due to
preciser actions used, the profit with precision of 0.05 is
higher than the profit with precision of 0.1, but the running
time of our RetailNet++ only takes less than 10 minutes

5. Related Work

Prior works, for example, (Nahmias, 1982), (Tsiros & Heil-
man, 2005), and (Li et al., 2009), on replenishing and pricing perishable products typically assume that the inventory is consumed in a first-in-first-out (FIFO) or last-in-first-out (LIFO) manner; both fresh and old products are priced the same; and/or the lifetime is two periods. Those works mainly focus on heuristic policies because it is difficult to solve the problem optimally. The most related work to ours

is (Ferguson & Koenigsberg, 2007), which considers the competition between fresh and old products with utility-maximizing consumers like our work but in a two-period horizon setting. However, retailers make strategic decisions in a long-term planning horizon. In addition, (Meadowcroft, 2016) shows that product display affects consumers purchasing behavior using a field experiment. We contribute this line of literature on perishable products by considering the intertwined effects of product display setting, discounting the old products, and replenishing the inventory. Artier studies on deteriorating products mainly focus on inventory replenishment policy and conclude that determining the optimal policy, even under simple modeling assumptions, is challenging. The authors in (Nahmias & Pierskalla, 1973) assume that a perishable product whose utility does not remain constant over time has a life of two periods, and the best policy can be achieved for the retailer by always ordering up to a constant level (Nahmias, 1975b)(Nahmias, 1975a)(Deuermeyer, 1980). This is usually referred to as the Order-Up-To inventory model. Besides, these studies conclude that if items perish in the same sequence as they are ordered, the results on fixed lifetime models hold for even stochastic lifetime variants (Ishii et al., 1981).

The dynamic replenishment policy involves competition between the vertically differentiated fresh products and old products. In (Ferguson & Koenigsberg, 2007), authors study the joint inventory and pricing decision of a deteriorating product in a two-period setting. The retailer can decide to carry all, some or none of the old products into the second period. The fresh products and old products co-exist, and in every period, the retailer makes decisions on the prices of both products and the order quantity of the fresh product. The authors claim that regardless of the pricing decision, the optimal price of the fresh products is the same in both periods. Authors in (Ferguson & Koenigsberg, 2007) claim that selling the old products in the second period exacerbate the detrimental effects of competition. In a similar setting, a joint inventory and pricing problem is studied for a retailer selling a product with a shelf life of two periods (Sainathan, 2013). Similar to our work, the fresh products and old products compete in their qualities and prices, and each customer selects the utility-maximizing product. The authors conclude that the benefit of selling the old products with consistent pricing and order decisions over all periods is much higher than the benefit from varying all the decisions in all the periods.

Reinforcement learning has been widely used to address the dynamic pricing problems since it can be formulated as an MDP framework. With the success of Atari games and board games, several deep-learning-based RL algorithms have been shown to have generalization and learning abilities (Mnih et al., 2013; 2015).
RetailNet: Enhancing Retail of Perishable Products with Multiple Selling Strategies via Pair-Wise Multi-Q network

(a) $N \sim U[0, 4]$ with parameter setting of Group 1

(b) $N \sim U[0, 4]$ with parameter setting of Group 2

(c) $N \sim B(1.0, 1.0)$ with parameter setting of Group 1

(d) $N \sim B(0.5, 1.0)$ with parameter setting of Group 2

Figure 3. Learning curves under different customer distributions and parameter settings

Sutton *et al.* (Sutton *et al.*, 2000) first introduces policy gradient when dealing with large-scale MDP problems. They propose a learnable function to approximate the $Q$ value to estimate the expected reward. To combine it with deep learning and improve the action exploration, (Mnih *et al.*, 2016) introduce asynchronous advantage actor-critic algorithm to create multiple agents and environments for each one and update global parameters asynchronously with just a few CPUs instead of GPUs. Driven by the issue of the inaccurate value function estimation, (Schulman *et al.*, 2015) propose Generalized Advantage Estimation (GAE) by multi-step iterations with a discount factor.

Some recent works focus on how to model the relation between multiple actions. A regularization term with covariance matrix is exploited to model the relation between different tasks (Wang & Yu, 2016; Zhang & Yeung, 2014). To solve the ranking problem, (He *et al.*, 2015) propose to use two separate networks to model the state and actions, respectively. Unlike their models, our proposed Multi-Q network not only can model each state-action pair together but also can capture the interdependence among the actions.

6. Conclusion and Future Work

We proposed RetailNet/RetailNet++ for dynamic multiple selling strategies in the retailing system to enhance long-term average profit. Our proposed RetailNet++ with Pair-wise Multi-Q network is capable of modeling each state-action pair and capturing the interdependence among the actions for an accurate value estimation. Experimentally, RetailNet/RetailNet++ produces near-optimal solutions efficiently, and can solve retailing problems with large-scale state and action spaces, where the traditional DP method cannot solve in reasonable time. In the future, we plan to explore more variants of Pair-wise Multi-Q network with self-attention on multi-action and multi-agent problems in retailing systems.
References


