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

Xiyao Ma [*1] **Fan Lu** [*1] **Xiajun Amy 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 Perishable products lose value over time and become outdated at the end of their shelf lives.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 are 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 are involved 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++ approaches the optimal profit with much shorter time. The

[*]Equal contribution [1]National Science Foundation Center for Big Learning, Large-scale Intelligent Systems Laboratory, Department of Electrical and Computer Engineering, University of Florida, Gainesville, Florida, 32611-6250, USA [2]Department of Information Systems and Operations Management, University of Florida, Gainesville, Florida, 32611-6250, USA. Correspondence to: Xiaolin Li <andyli@ece.ufl.edu>, Xiyao Ma <maxiy@ufl.edu>.

Reinforcement Learning for Real Life (RL4RealLife) Workshop in the 36th International Conference on Machine Learning, Long Beach, California, USA, 2019.

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_h and the freshness level drops to Q_O 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 (α) for one unit of freshness level. We assume α is uniformly distributed over $[0, \overline{\alpha}]$ as in Ferguson & Koenigsberg (2007). A consumer with valuation α derives a utility of $U_F(\alpha) = \alpha Q_f - p$ from consuming a fresh product and $U_O(\alpha) = \alpha Q_O - p(1-\rho)$ 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 ρ.

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.

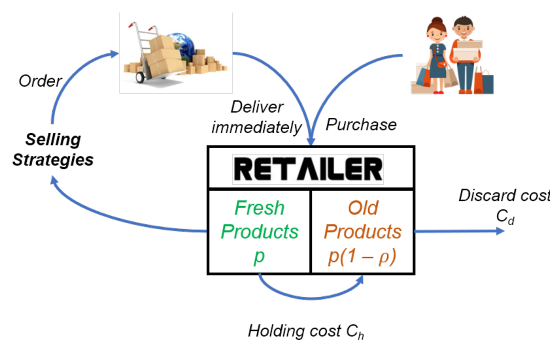


Figure 1. Illustration of the Retailing System

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;

Discount	ρ range	P_O	P_F	P_{OF}	P_{FO}
very low	$[0, 1-\frac{Q_O\overline{\alpha}}{p})$	0	$1-F(\frac{p}{Q_f})$	0	0
Low	$[1-\frac{Q_O\overline{\alpha}}{p}, \frac{Q_f-Q_O}{Q_f})$	0	$F(\frac{p(1-\rho_p)}{Q_O})-F(\frac{p}{Q_f})$	0	$1-F(\frac{p(1-\rho_p)}{Q_O})$
Medium	$[\frac{Q_f-Q_O}{Q_f}, \frac{Q_f-Q_O}{p}\overline{\alpha})$	$F(\frac{p}{Q_f})-F(\frac{p(1-\rho_p)}{Q_O})$	0	$F(\frac{p\rho_p}{Q_f-Q_O})-F(\frac{p}{Q_f})$	$1-F(\frac{p\rho_p}{Q_f-Q_O})$
High	$[\frac{Q_f-Q_O}{p}\overline{\alpha}, 1)$	$F(\frac{p}{Q_f})-F(\frac{p(1-\rho_p)}{Q_O})$	0	$1-F(\frac{p}{Q_f})$	0

Table 1. P_O, P_F, P_{OF}, P_{OF} calculations as a function F of the discount parameter ρ, where F follows a uniform distribution.

- Setting C: Old products are displayed on the shelves only after fresh products run out such that old products are not available to consumers before the fresh products run out;

- Setting $C^{'}$: Fresh products are displayed on the shelves only after the old products run out such that fresh products are not available to consumers before the old products run out.

We further distinguish consumers in terms of searching behavior: active and passive. Active consumers look for the product which maximizes their utility despite of product display locations. In contrast, passive consumers only consider products easier to reach, i.e., the front of the shelves. Let β and $1-\beta$ denote the proportion of passive and active consumers, respectively. We use F_0 and O_0 to denote the proportions of consumers purchasing the fresh and old products when both products are in stock, respectively, which are calculated as shown in Table 2.

	F_0	O_0
A	$P_F + P_{FO}$	$P_O + P_{OF}$
B	$(1-\beta)P_F + (1-\beta)P_{FO}$	$P_O + P_{OF} + \beta P_{FO}$
$B^{'}$	$P_F + P_{FO} + \beta P_{OF}$	$(1-\beta)P_O + (1-\beta)P_{OF}$
C	$P_F + P_{FO} + P_{OF}$	0
$C^{'}$	0	$P_O + P_{FO} + P_{OF}$

Table 2. Consumer choice in all five product display settings

In Setting A, since both products are displayed in front, active and passive consumers have the same purchasing behavior, i.e., choosing the one maximizing their utility. In setting B, only active consumers of type F, FO purchase the fresh products as the passive consumers only consider the old product displayed in front, while all consumers of type O, OF and passive consumers of type FO purchase the old products. The same logic applies to Setting $B^{'}$. In Setting C ($C^{'}$), before the fresh (old) products run out, only fresh (old) products are displayed on the shelves so that both active and passive consumers have the same purchasing behavior.

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

$$\begin{aligned}\Pi(y_t; I, N) =& pS_F(y_t; I, N) + p(1-\rho)S_O(y_t; I, N) \\ & - C_d(I - S_O(y_t; I, N)) - C_f y_t - C_h I \\ & = pS_F(y_t; I, N) + (p(1-\rho) + C_d) \\ & + S_O(y_t; I, N) - C_f y_t - (C_h + C_d)I\end{aligned} \tag{1}$$

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 totaol quantity of customers N. The detail of computing S_F and S_O is out of scope. However, we provide the link [1] 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 (ρ_t). The profit-maximization problem is as follows.

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

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.

$$\max_{y_1,\ldots,y_T \geq 0} \frac{1}{T}\sum_{t=1}^{T} E_{N_t}[\Pi(y_t)] \tag{3}$$

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 γ, given Bellman Equation(Bellman, 1957):

$$R_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{4}$$

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 θ_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 θ_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:

$$h = W_h(s) + b_h, \tag{5}$$

$$a = Softmax(W_a h + b_a), \tag{6}$$

$$v = W_v h + b_v. \tag{7}$$

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 ρ_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^i) state-action pair together and capture the interdependence among the actions. As shown in Fig. 2(b), we use Multi-Q network $q_i = Q_i(s_t, a_t^i)$ 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.

$$Q_{final}(s_t, a_t) = \sum_{i=1}^{M} \phi_i q_i = \Phi^T Q \tag{8}$$

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

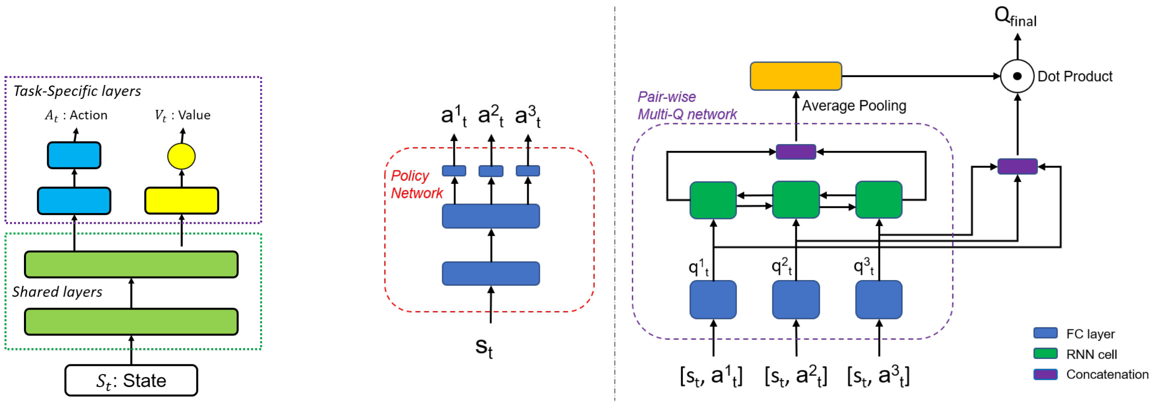
Then we exploit a bi-directional GRU network to model Φ:

$$\Phi = BiGRU(q_i, h_i) \tag{9}$$

where h_i is the current hidden state. We further obtain the final weight Φ 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



(a) RetailNet Architecture

(b) RetailNet++: Architecture of pair-wise Multi-Q network.

Figure 2. Architecture of RetailNet and RetailNet++: (a) RetailNet: With shared lower layers and different higher layers networks, the policy and value network generate action a_t and value estimation V_t on the current state s_t. (b) RetailNet++: Use multiple pair-wise Q-network to obtain multiple Q values by capturing the specific relationship between each state-action pair. The weights modeling the interdependence between multiple Q value with Bi-directional RNN are used to compute the Q_{final} value via inner product.

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:

$$\begin{aligned} \hat{A}_t^{GAE(\gamma,\lambda)} &= \delta_t + (\gamma\lambda)\delta_{t+1} + ... + (\gamma\lambda)^{T-t+1}\delta_{T-1} \\ &= \sum_{l=0}^{\infty}(\gamma\lambda)^l \delta_{t+l}^V \qquad (10) \\ \delta_t^V &= r_t + \gamma V(s_{t+1};\theta_v) - V(s_t;\theta_v) \qquad (11) \end{aligned}$$

where λ 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):

$$H(\pi(a_t^i|s_t;\theta_a)) = -\log\pi(a_t^i|s_t;\theta_a)\pi(a_t^i|s_t;\theta_a) \qquad (12)$$

Therefore, we have a total policy loss function for updating the parameters θ_a of the policy network:

$$\begin{aligned} \mathcal{L}(\theta_a) = &\ -\hat{A}_t^{GAE(\gamma,\lambda)} - \mu H(\pi(a_t^i|s_t;\theta_a)) \qquad (13) \\ = &\ -\textstyle\sum_{l=0}^{\infty}\log\pi(a_i|S_t;\theta_a)(\gamma\lambda)^l\delta_{t+l}^V \\ &\ -\mu\log\pi(a_i|S_t;\theta_a)\pi(a_i|S_t;\theta_a) \qquad (14) \end{aligned}$$

where μ controls the magnitude of entropy for action a_i.

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

$$\mathcal{L}(\theta_v) = 0.5E(R_t - V(s;\theta_v))^2 \qquad (15)$$

As shown in Algorithm 1, we train our proposed RetailNet/RetailNet++ with A3C algorithm in an asynchronous update manner. The global shared parameters θ_a and θ_v are updated with thread-specific gradient θ'_a and θ'_v when the thread episode terminates, and update thread-specific parameters θ'_a and θ'_v by copying global parameters θ_a and θ_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.

	p	C_f	C_h	C_d	Q_f	Q_O	$MaxN$
Group 1	4.0	0.5	0.10	0.2	10.0	4.0	4.0
Group 2	3.5	0.3	0.05	0.1	8.0	3.5	10.0

Table 3. Parameter settings across all experiments

The two groups of parameters shown in Table 3 are used

Algorithm 1 Training RetailNet with Asynchronous Update

Initialize global shared parameters θ_a and θ_v and global step counter T = 0
Initialize local parameters θ_a', θ_v', local step counter t = 0 and t_{end}
repeat
  Reset global shared parameters gradients: $d\theta_a \leftarrow 0$ $d\theta_v \leftarrow 0$
  Update local parameters $\theta_a' = \theta_a$, $\theta_v' = \theta_v$
  Get current state s_t
  repeat
    Execute a_t^i based on Replenish Policy Network $\pi(a_t^i|s_t;\theta')$
    Compute $Q_{final}(s_t, a_t)$ via pair-wise Multi-Q networks given $[s_t, a_t^i]$
    Interact with retailing system simulator and receive immediate reward r_t and enter into next state s_{t+1}
    $t \leftarrow t+1$ and $T \leftarrow T+1$
  until Terminate($t = t_{end}$)
  $R = \begin{cases} \mathrm{P}_t - C_d(y_t - S_F(y_t; I, N)) & \text{for terminal } s_t \\ \mathrm{P}_t & \text{for non-terminal } s_t \end{cases}$
  for $i \in [0, ..., t_{end} - 1]$ **do**
    $R = \gamma R + r_i$
    Accumulate gradients w.r.t θ_a':
    $d\theta_a \leftarrow d\theta_a + \nabla_{\theta_a'} log\pi(a_t^i|s_i;\theta_a')((\gamma\,\lambda)^l \delta_{t+l}^V) + \sigma\nabla_{\theta_a'} H(\pi(a_t^i|s_t;\theta_a))$
    Accumulate gradients w.r.t θ_v':
    $d\theta_v \leftarrow d\theta_v + \partial(R - V(s_i;\theta_v'))^2/\partial\theta_v'$
  end for
  Asynchronous update θ_a and θ_v with $d\theta_a$ and $d\theta_v$
until $T > T_{max}$

across experiments. Values of parameters are selected based on consumers' purchasing behavior, as described in Table 1. Essentially, a consumer purchases the product that gives the highest positive utility or nothing if no products provide a positive utility.

We use the Profit Gap as the the performance measurement: Profit Gap $= \frac{\text{optProfit}-\text{Profit}_\text{R}}{\text{optProfit}} \times 100\%$, where Profit_R is the profit achieved by our proposed RetailNets, and 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 [2]. 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 realize 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.

Models	Profit Gap (%)	Profit$_R$	optProfit	Time (min)
RetailNet($d = A$, $\rho = 0.6$)	0	8.400	8.400	3.21
RetailNet($d = B$, $\rho = 0.4$)	0	7.296	7.296	4.38
RetailNet($d = B^{'}$, $\rho = 0.5$)	0	2.800	2.800	5.12
RetailNet($d = C$, $\rho = 0.7$)	0	8.640	8.640	4.13
RetailNet($d = C^{'}$, $\rho = 0.9$)	0	8.400	8.400	3.85
RetailNet++	0	8.640	8.640	16.99
RetailNet($d = A$, $\rho = 0.6$)	0.0050	18.2992	18.2993	5.12
RetailNet($d = B$, $\rho = 0.5$)	0	16.2500	16.2500	4.37
RetailNet($d = B^{'}$, $\rho = 0.5$)	0	7.0000	7.0000	3.56
RetailNet($d = C$, $\rho = 0.7$)	0.0256	19.5200	19.5250	6.43
RetailNet($d = C^{'}$, $\rho = 0.9$)	0	18.5625	18.5625	5.74
RetailNet++	NA	19.5200	NA	28.47

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

[2] Available at: https://sites.google.com/view/retailnet0/appendix

Models	Profit Gap (%)	Profit$_R$	Profit$_{DP}$	Time (min)
RetailNet($d = A$, ρ=0.7)	0.8486	3.4702	3.4999	14.17/328.09
RetailNet($d = B$, ρ=0.5)	0.0489	3.2734	3.2750	18.56/330.12
RetailNet($d = B'$, ρ=0.5)	1.0641	2.7705	2.8003	15.97/325.18
RetailNet($d = C$, ρ=0.8)	0.1156	3.7167	**3.7210**	21.06/355.76
RetailNet($d = C'$, ρ=0.5)	0.2283	3.4960	3.5040	19.11/347.82
RetailNet++ (0.1)	NA	**3.7233**	NA	24.19/NA
RetailNet++ (0.05)	NA	**3.7324**	NA	30.48/NA
Baseline 1 (0.05)	NA	3.6788	NA	30.48/NA
Baseline 2 (0.05)	NA	3.7218	NA	30.48/NA
RetailNet($d = A$, ρ=0.7)	0.2897	7.7790	7.8016	16.36/328.09
RetailNet($d = B$, ρ=0.5)	0.4110	7.3898	7.4203	19.51/330.12
RetailNet($d = B'$, ρ=0.5)	0.6784	6.2514	6.2941	15.53/325.18
RetailNet($d = C$, ρ=0.8)	1.2982	8.5075	**8.6194**	21.30/355.76
RetailNet($d = C'$, ρ=0.5)	1.0545	7.9010	7.9852	18.58/347.82
RetailNet++ (0.1)	NA	**8.6269**	NA	27.60/NA
RetailNet++ (0.05)	NA	**8.6328**	NA	30.30/NA
Baseline 1 (0.05)	NA	8.3449	NA	30.48/NA
Baseline 2 (0.05)	NA	8.6149	NA	30.48/NA

Table 5. Profit and time by RetailNet and RetailNet++ with customer 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.

Models	Profit Gap (%)	Profit$_R$	Profit$_{DP}$	Time (min)
RetailNet($d = A$, ρ=0.6)	0.0308	4.9588	4.9603	8.59/394.04
RetailNet($d = B$, ρ=0.6)	0.1182	4.6107	4.6161	9.63/383.05
RetailNet($d = B'$, ρ=0.6)	0.2471	4.1579	4.1682	7.42/253.98
RetailNet($d = C$, ρ=0.8)	0.1757	5.2633	**5.2726**	9.12/396.49
RetailNet($d = C'$, ρ=0.5)	0.0552	4.9208	4.9235	7.55/262.63
RetailNet++ (0.1)	NA	**5.3067**	NA	16.50/NA
RetailNet++ (0.05)	NA	**5.5158**	NA	24.83/NA
Baseline 1 (0.05)	NA	5.2408	NA	24.83/NA
Baseline 2 (0.05)	NA	5.4933	NA	24.83/NA
RetailNet($d = A$, ρ=0.6)	0.7958	11.0953	11.1840	28.12/724.13
RetailNet($d = B$, ρ=0.5)	0.0492	10.3586	10.3637	27.94/722.95
RetailNet($d = B'$, ρ=0.5)	0.1073	9.3090	9.3190	20.47/555.80
RetailNet($d = C$, ρ=0.8)	0.7595	11.7986	**11.8889**	19.73/779.76
RetailNet($d = C'$, ρ=0.5)	0.1982	11.079	11.101	22.96/707.84
RetailNet++ (0.1)	NA	**11.9668**	NA	19.57/NA
RetailNet++ (0.05)	NA	**11.9735**	NA	27.17/NA
Baseline 1 (0.05)	NA	11.5435	NA	24.83/NA
Baseline 2 (0.05)	NA	11.8059	NA	24.83/NA

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

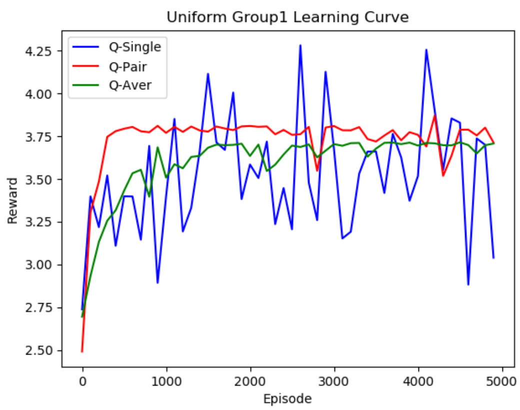
0.05 respectively. As shown in Table 5 and Table 6, due to more precise actions used, the profit with precision of 0.05 is higher than the profit with precision of 0.1, but the running time of our RetailNet++ only takes less than 10 minutes more.

5. Related Work

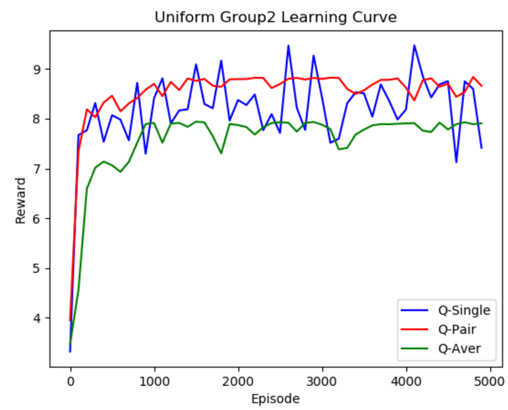
Prior works, for example, (Nahmias, 1982), (Tsiros & Heilman, 2005), and (Li et al., 2009), on replenishing and pricing perishable products typically assume that the inventory is consumed in a first-in-first-out (FIFO) or last-in-first-out (LIFO) manner; both fresh and old products are priced the same; 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. arlier 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.

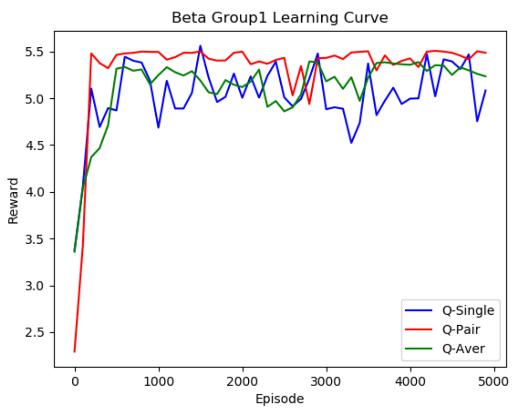
Reinforcement learning has been widely used to address the dynamic pricing problems since it can be formulated as an MDP framework. With the success of Atari games and board games, several deep-learning-based RL algorithms have been shown to have generalization and learning abilities (Mnih et al., 2013; 2015).



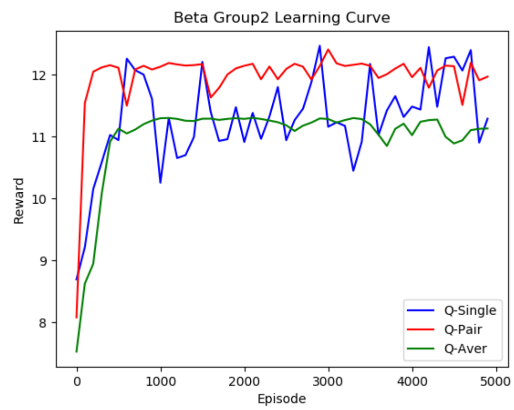
(a) $N \sim U[0, 4]$ with parameter setting of Group 1



(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

Bellman, R. Dynamic programming. *Princeton, USA: Princeton University Press*, 1(2):3, 1957.

Deuermeyer, B. L. A single period model for a multiproduct perishable inventory system with economic substitution. *Naval Research Logistics Quarterly*, 27(2):177–185, 1980.

Ferguson, M. E. and Koenigsberg, O. How should a firm manage deteriorating inventory? *Production and Operations Management*, 16(3):306–321, 2007.

Goyal, S. and Giri, B. C. Recent trends in modeling of deteriorating inventory. *European Journal of operational research*, 134(1):1–16, 2001.

He, J., Chen, J., He, X., Gao, J., Li, L., Deng, L., and Ostendorf, M. Deep reinforcement learning with a natural language action space. *arXiv preprint arXiv:1511.04636*, 2015.

Honhon, D. and Seshadri, S. Fixed vs. random proportions demand models for the assortment planning problem under stockout-based substitution. *Manufacturing & Service Operations Management*, 15(3):378–386, 2013.

Ishii, H., Nose, T., Shiode, S., and Nishida, T. Perishable inventory management subject to stochastic leadtime. *European Journal of Operational Research*, 8(1):76–85, 1981.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, Y., Lim, A., and Rodrigues, B. Notepricing and inventory control for a perishable product. *Manufacturing & Service Operations Management*, 11(3):538–542, 2009.

Li, Y., Kang, H., Ye, K., Yin, S., and Li, X. Foldingzero: Protein folding from scratch in hydrophobic-polar model. *arXiv preprint arXiv:1812.00967*, 2018.

Meadowcroft, D. *Understanding the effect of product displays on consumer choice and food waste: a field experiment*. PhD thesis, University of Delaware, 2016.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Nahmias, S. A comparison of alternative approximations for ordering perishable inventory. *INFOR: Information Systems and Operational Research*, 13(2):175–184, 1975a.

Nahmias, S. Optimal ordering policies for perishable inventoryii. *Operations Research*, 23(4):735–749, 1975b.

Nahmias, S. Perishable inventory theory: A review. *Operations research*, 30(4):680–708, 1982.

Nahmias, S. and Pierskalla, W. P. Optimal ordering policies for a product that perishes in two periods subject to stochastic demand. *Naval Research Logistics Quarterly*, 20(2):207–229, 1973.

Raju, C., Narahari, Y., and Ravikumar, K. Reinforcement learning applications in dynamic pricing of retail markets. In *E-Commerce, 2003. CEC 2003. IEEE International Conference on*, pp. 339–346. IEEE, 2003.

Sainathan, A. Pricing and replenishment of competing perishable product variants under dynamic demand substitution. *Production and Operations Management*, 22 (5):1157–1181, 2013.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Sutton, R. S., Barto, A. G., et al. *Reinforcement learning: An introduction*. MIT press, 1998.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Tsiros, M. and Heilman, C. M. The effect of expiration dates and perceived risk on purchasing behavior in grocery store perishable categories. *Journal of marketing*, 69(2): 114–129, 2005.

Wang, H. and Yu, Y. Exploring multi-action relationship in reinforcement learning. In *Pacific Rim International Conference on Artificial Intelligence*, pp. 574–587. Springer, 2016.

Watkins, C. J. C. H. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

Zhang, Y. and Yeung, D.-Y. A regularization approach to learning task relationships in multitask learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(3):12, 2014.