INVENTORY MANAGEMENT WITH Q-LEARNING AND DEEP Q-NETWORKS

CS 238 FINAL PROJECT

Zachary A. Phelps
zphelps@stanford.edu

Cristian Tavarez
ctavarez@stanford.edu

December 9, 2024

ABSTRACT

Inventory management is a fundamental problem in supply chain optimization, involving decisions on ordering and maintaining stock levels to maximize profitability while minimizing costs. This paper aims to address the single-product inventory management problem under uncertain demand using reinforcement learning techniques, specifically Q-learning, and Deep Q-Networks (DQN). The proposed approach is compared with the traditional (s, S) policy, which establishes a reorder point s and target inventory levels S. We present the problem as a Markov Decision Process (MDP), discuss all approaches, conduct extensive experiments to evaluate the results, and present insights into the efficacy of using reinforcement learning for inventory optimization.

1 Introduction

Inventory management plays a critical role in supply chain and logistics. The goal is to ensure that a retail store maintains optimal stock levels to meet uncertain customer demand without incurring excessive holding costs or facing stockouts. Traditionally, inventory management problems have been solved using operations research techniques such as dynamic programming, (s, S) policies, and Economic Order Quantity (EOQ) models. However, these approaches often assume perfect knowledge of demand distributions or require significant computational resources when dealing with large state spaces.

In recent years, reinforcement learning (RL) techniques have emerged as promising tools for decision-making under uncertainty, especially when demand is difficult to model precisely. Among these techniques, both Q-learning and Deep Q-Networks (DQNs) offer model-free approaches that can learn optimal policies without requiring explicit knowledge of the demand distribution. This paper presents inventory management systems using both Q-learning and DQN to address the challenge of maximizing cumulative profit. We compare their performance against a simple baseline policy that maintains a target inventory level, illustrating how these RL methods can dynamically respond to changing demand patterns and large state spaces.

2 Related Work

Inventory management has been extensively studied within the field of operations research, leading to the development of various classical methods aimed at optimizing stock levels. One of the earliest and most widely used approaches is the Economic Order Quantity (EOQ) model [1] which provides a closed-form solution for determining the optimal order quantity that minimizes the total cost, including ordering and holding costs. However, the EOQ model assumes constant consumer demand DS, which limits its applicability in environments where demand is uncertain and fluctuates over time.

Another traditional approach is the (s, S) [2] policy, which establishes a reorder point s and target inventory levels S. When inventory drops below the reorder point s, an order is placed to replenish inventory up to level S. This policy works well under certain stochastic conditions, but it requires accurate estimation of parameters and assumes a stationary demand distribution, which may not hold in practice.

Recent advances in machine learning have allowed reinforcement learning (RL) techniques to be applied to inventory management. One of those advances has been Q-learning, introduced by Watkins [3], Q-learning is a model-free RL algorithm that operates by estimating the value of taking a particular action in a given state and updating these estimates iteratively using a reward-based learning mechanism. In the context of inventory management, Q-learning allows the agent to learn an optimal ordering policy by interacting with the environment and adjusting its decisions based on the observed outcomes and rewards from the action taken, without needing explicit prior knowledge of demand patterns.

Another breakthrough was the introduction of Deep Q-Networks by Mnih *et al.* [4], which combines Q-learning with deep neural networks to enable agents to learn optimal policies directly from high-dimensional input data. DQN showed human-level performance in playing Atari games by using raw pixels as input and approximating Q-values through a convolutional neural network. Knowing this, DQN can then be used in the context of single item inventory management to optimize ordering policies in environments with complex or uncertain demand patterns approximating Q-values for state-action pairs.

3 Problem Formulation

We consider a single-product retail store that must manage inventory over a sequence of discrete time steps (days). At each time step t, the store must decide how many units of the product to order to meet future demand, which is stochastic and follows a Poisson distribution. We assume that the orders placed at time step t will be delivered by time step $t+1$. Costs include ordering costs and inventory holding costs. The objective is to make sequential replenishment decisions to maximize expected cumulative discounted profit over a year-long time horizon (365 steps/days).

We have modeled this problem as the following MDP $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$:

- State space $\mathcal{S}$: The current inventory level for the retail store, where $I_{\texttt{max}}$ is the maximum inventory capacity, and the current day of the week.

$$\mathcal{S} = \{(i, w) \mid i \in \{0, 1, 2, \ldots, I_{\texttt{max}}\}, w \in \{0, 1, 2, 3, 4, 5, 6\}\} \tag{1}$$

  By including w, we allow the model to incorporate day-dependent demand patterns. For example, weekdays ($w < 5$) may have lower average demand, while weekends ($w \geq 5$) may have higher demand.
- Action space $\mathcal{A} = \{0, 1, 2, \ldots, Q_{\texttt{max}}\}$: The quantity of the order to be placed for delivery in the next time step $(t+1)$, where $Q_{\texttt{max}}$ is the maximum order quantity allowed.
- Transition function $\mathcal{T}((i_{t+1}, w_{t+1}) \mid (i_t, w_t), a_t)$: After taking action a_t at state (i_t, w_t), we transition to a new state (i_{t+1}, w_{t+1}) on the next day. Demand on day t is sampled from a Poisson distribution whose parameters depends on the day of the week:

$$D_t = \begin{cases} \text{Poisson}(\lambda_{weekday}) & w < 5 \\ \text{Poisson}(\lambda_{weekend}) & w \geq 5 \end{cases} \tag{2}$$

  Therefore, the inventory at the beginning of day $i+1$ is given by:

$$i_{t+1} = \max\{i_t + a_t - D_t, 0\} \tag{3}$$

  and the day of the week advances by one:

$$w_{t+1} = (w+1) \mod 7. \tag{4}$$

  Since D_t is stochastic, the transition probabilities depend on the Poisson distribution with the appropriate parameters $\lambda_{weekday}$ or $\lambda_{weekend}$.
- Reward function $\mathcal{R}((i_t, w_t), a_t)$: The net profit earned at time t. Specifically, net profit is the revenue from sales minus the ordering and holding costs. Let p be the selling price per unit, c_o be the cost of ordering per unit, and c_h be the cost of holding per unit. Sales on day t are limited by the minimum of available units $(i_t + a_t)$ and the demand D_t, thus the Revenue is:

$$\text{Revenue} = p \cdot \min(i_t + a_t, D_t).$$

  Ordering costs equals the number of units ordered time price:

$$\text{Ordering Cost} = c_o \cdot a_t.$$

  At the end of the day, the leftover inventory is whatever supply is left:

$$i_{t+1} = \max(i_t + a_t - D_t, 0),$$

  Therefore, our reward function is

$$R((i_t, w_t), a_t) = p \cdot \min(i_t + a_t, D_t) - c_o \cdot a_t - c_h \cdot i_{t+1}.$$

3.1 Environment Setup Parameters

- $I_{max} = 50$
- $Q_{max} = 20$
- $\lambda_{weekday} = 10$
- $\lambda_{weekend} = 20$
- $p = 50$
- $c_h = 3$
- $c_o = 30$

4 Approach

This section introduces and compares three approaches to inventory management: a Baseline Policy, Q-Learning, and Deep Q-Networks (DQNs). Each approach is described below.

4.1 Baseline Policy

The baseline policy implemented for comparison is the traditional (s, S) inventory control policy. This policy operates by maintaining inventory levels between a predefined reorder point s and a target maximum inventory level S. Specifically, when the inventory level at the start of a time step falls below s, the policy triggers an order to replenish stock back to S. This simple, rule-based policy serves as a benchmark to evaluate the adaptive capabilities of learning-based approaches. While easy to implement, the baseline policy does not account for dynamic demand fluctuations or long-term profitability.

4.2 Q-Learning

Q-learning is a model-free reinforcement learning technique that iteratively learns an optimal policy by interacting with the environment and updating state-action value estimates storing the values in a table. Unlike traditional analytical approaches, such as (s, S) policies that rely on known demand distributions and cost parameters, Q-Learning does not require explicit modeling of the underlying stochastic processes. Instead, it uses trial-and-error exploration to adjust its value function, gradually improving its ordering decisions over time. While this flexibility allows Q-Learning to handle more complex and uncertain environments, it may perform worse than a well-tuned (s,S) policy in stable, fully understood settings.

Q-Learning achieves this by using an update rule derived from the Bellman equation. After observing a transition from state s to state s' by taking action a, receiving reward r, and then selecting a future action a', the Q-value is updated as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big],$$

where α is the learning rate, γ is the discount factor, and $Q(s,a)$ is the state-action value function.

4.3 Q-Learning Hyper-parameters

- $\gamma = 0.90$
- $\varepsilon = 0.1$
- $\alpha = 0.01$

4.4 Deep Q-network (DQN)

Deep Q-Networks (DQNs) extend the Q-Learning framework by utilizing a deep neural network to approximate the state-action value function $Q(s,a)$. This allows DQNs to handle complex state spaces more efficiently than table-based methods. During training, DQNs rely on stochastic gradient descent to iteratively adjust the network parameters θ and minimize the TD error:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim U(\mathcal{D})}\left[(y - Q(s,a;\theta))^2\right],$$

where

$$y = r + \gamma \max_{a'} Q(s',a';\theta^-),$$

and θ^- are the parameters of a fixed target network periodically updated to stabilize learning. By sampling mini-batches of past experiences (s, a, r, s') from a replay memory $\mathcal{D}$, DQN breaks the temporal correlations in the training data and reuses past experience for more efficient learning. This combination of neural function approximation, stochastic gradient updates, and experience replay allows DQNs to converge to higher cumulative rewards than both the baseline (s, S) policy and standard Q-Learning.

4.5 DQN Agent Hyper-parameters

- $\gamma = 0.99$
- $\varepsilon_{start} = 1$
- $\varepsilon_{end} = 0.01$
- $\varepsilon_{decay} = 0.995$
- $\alpha = 0.001$
- Target Update $\tau = 10$
- Memory Size $= 10000$

5 Results

We first ran a algorithm for our (s, S) policy which assigned (s, S) values and simulated a calendar year (365 time steps) and kept track of which (s, S) values performed the best. This resulted in the optimal values of $(s, S) = (26, 32)$. Next, we trained the Q-Learning agent for 50,000 episodes. An episode is a simulation of the agent running the policy over 365-time steps to simulate a full year. As shown in Figure 1, the Q-Learning agent converged around the 15,000th episode, indicating that additional training was unnecessary.

Next, we trained our DQN agent for 5,000 episodes. Compared to Q-Learning, training the DQN agent required significantly more time per episode. As shown in Figure 5, the DQN agent initially suffered large negative profits, but its performance improved rapidly as it learned from each episode. Ultimately, it converged to an average profit of approximately \$70,000 per simulated year. These initial negative rewards and longer training times highlight that if training time is limited, Q-Learning may be the preferred approach due to its faster convergence and lower computational demands. However, both Q-Learning and DQN agents can be effectively fine-tuned with fewer training episodes, as evidenced by their convergence behaviors.

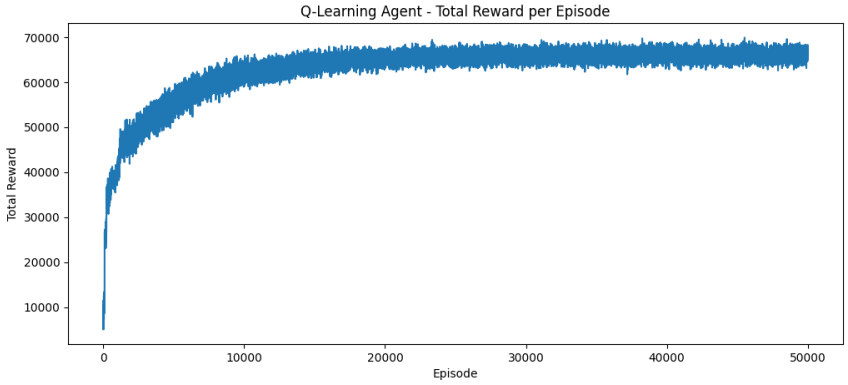

Figure 1: Q-Learning Training

After saving the optimal (s, S) policy values, the Q-Learning table, and the DQN neural network, we ran all three policies for the same simulated year to compare their performance. As shown in Figure 3, the baseline policy delivered the lowest profit at $61,399$, while the Q-Learning agent improved on this by achieving 66,199. The DQN agent outperformed both, achieving a profit of $69,602$, which aligns well with our training results.

These findings support our previous observations about the limitations of the (s, S) policy. Since (s, S) assumes perfect knowledge of the demand distribution and cannot adapt to changes in the environment, its performance suffers compared to learning-based approaches. In contrast, the Q-Learning agent demonstrated greater flexibility but faced challenges in

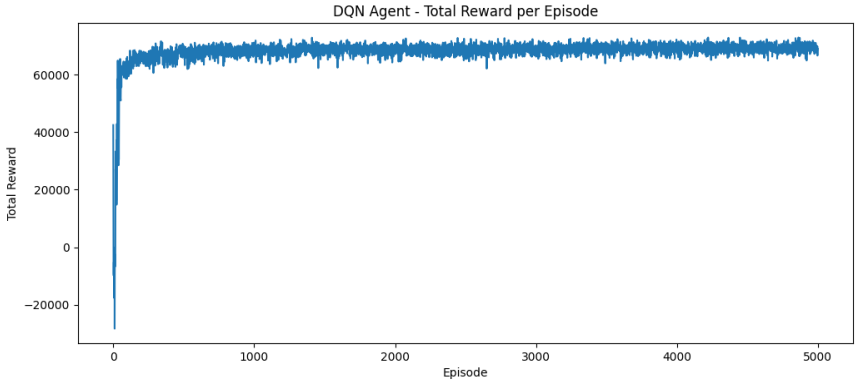

Figure 2: DQN Training

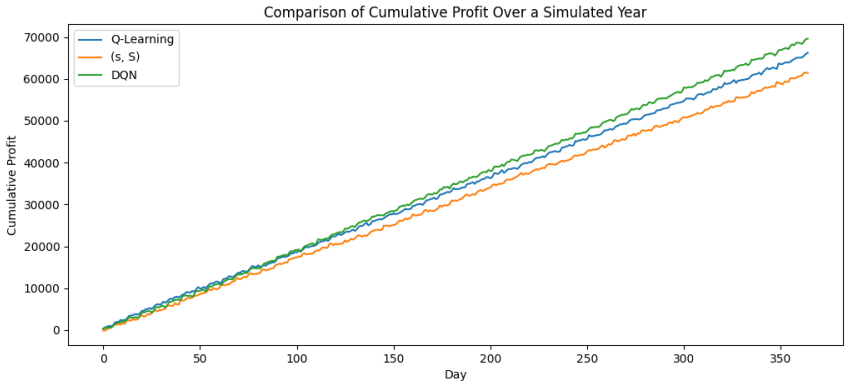

Figure 3: Evaluating Rewards Across (s, S) Policy, Q-Learning, and DQN Approaches

handling larger state spaces due to its reliance on tabular methods. The DQN agent's ability to approximate values across a vast range of state-action pairs allowed it to scale effectively and adapt to the uncertainties of our inventory management problem. These advantages make DQN particularly suited for problems with large state spaces and uncertain demand.

6 Conclusion

Our project has demonstrated the potential of reinforcement learning methods, specifically Q-learning and DQN, in addressing the inventory management problem where demand is uncertain and the state space is large. By comparing these approaches to a traditional (s, S) policy and a simple baseline strategy, we showed that both Q-learning and DQN can adapt to changing demand patterns without relying on explicit knowledge of the underlying distribution. The results highlighted the limitations of rigid, distribution-dependent policies and revealed that, while Q-learning improves on classical approaches, DQN's function approximation capabilities offer even greater scalability and robustness.

7 Limitations and Future Work

While effective for a single-item, instantaneous-replenishment setting, our approach can be extended to more realistic scenarios. Introducing factors like lead times which would change the shipment of items to no longer be instantaneous, but instead take x amount of days. This would force the agent to anticipate delays and manage order timing more strategically, increasing complexity. Additionally, moving from a single-item problem to a multi-item environment, where products interact through correlated demands and shared constraints, would better reflect real-world conditions. Future work could integrate these new complexities, employing advanced RL algorithms and leveraging the scalability of DQNs to handle larger state spaces, uncertain demand interactions, and delayed shipments.

8 Contributions

Both members of the group contributed equally to the project. We worked together to formulate the problem, discuss approaches, and define the MDP. Zach coded up the model and ran the simulations, while Cristian documented the results in our paper.

9 Acknowledgment

Thank you to Mykel Kochenderfer and the entire teaching staff for a fantastic quarter! We really enjoyed the class and learned a lot!

References

[1] D. Erlenkotter, "Ford Whitman Harris and the Economic Order Quantity Model," *Operations Research*, vol. 38, no. 6, pp. 937–946, 1990.

[2] Herbert E. Scarf. The optimality of (S, s) policies in the dynamic inventory problem. In *Management Science*, volume 6, number 1, pages 49–71. INFORMS, 1960.

[3] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3–4, pp. 279–292, 1992.

[4] Volodymyr Mnih *et al.* Human-level control through deep reinforcement learning. *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.