Can Q-learning Play Inventory Game?

PRESENTED BY THUREIN WIN HEIN
SUPERVISED BY PROF. HUYNH TRUNG LUONG

thureinwinhein2000@gmail.com

What is inventory Management?

- To meet demand on time, companies often keep on hand stock in inventory.

- Inventory Management answer the following questions.

1. When should an order be placed for a product?
2. How large should each order be?

Costs Involved in Inventory Management

Ordering and Setup Cost:

Expenses incurred each time an order is placed to supplier.

Unit Purchasing Cost:

The price paid per unit of a product when purchasing from a supplier.

Holding or Carrying Cost:

Expenses to store and maintain inventory (e.g., warehousing, insurance, obsolescence).

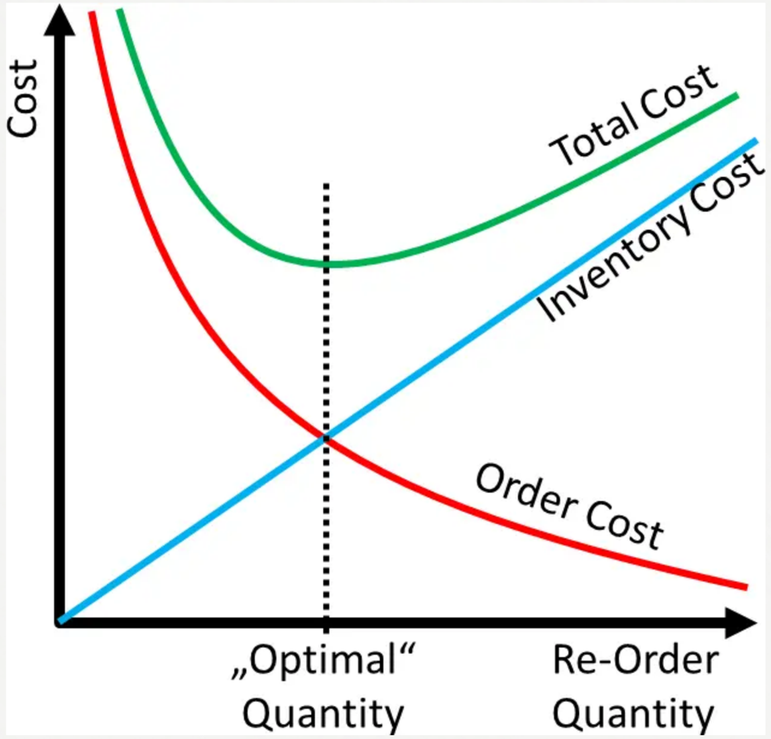
Stockout or Shortage Cost:

Losses from unmet demand, including lost sales, customer dissatisfaction

The above are just common examples, there are still otherof Inventory associated costs such as spoliage cost or inventory insspection cost.

What are the challenges?

	Ordering Too Late/Small	Ordering Too Early/Large
Holding Costs	Low	High
Stockout Risk	High	Low
Ordering Costs	High (frequent orders)	Low (less frequent orders)



- The goal of Inventory Game is to find the "sweet spot with lowest cost" between these extremes.

Stochastic and Complex Real World Inventory System

Traditional inventory models such as the economic order quantity (EOQ) model provide quick solutions assuming constant demand and complete information.

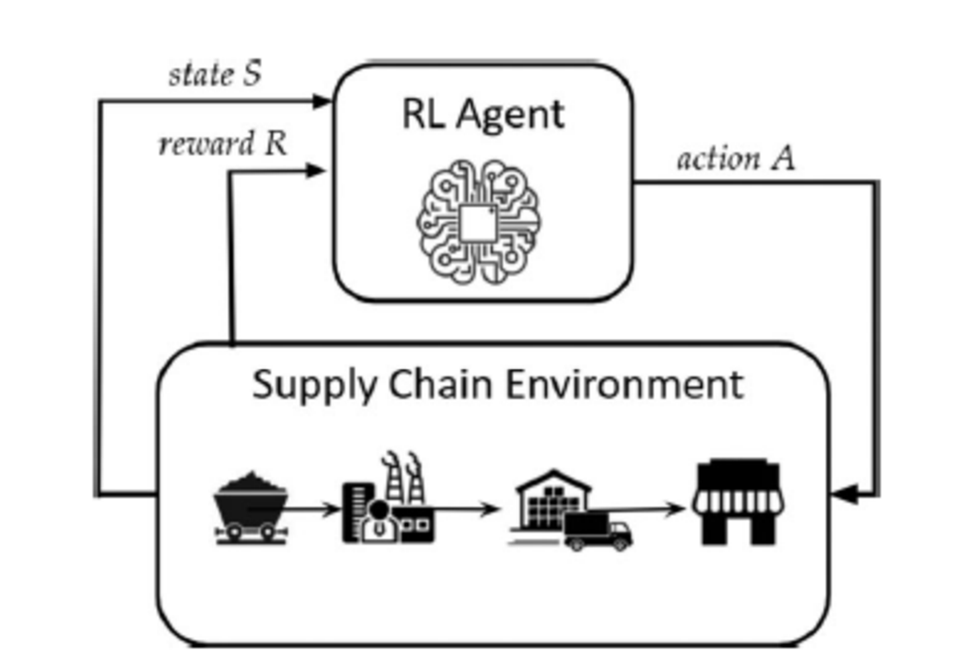
However, in real inventory system

- Demand is stochastic and have patterns and trends.
- Lead Time is uncertain.
- Incomplete Information.
- real world inventory complexity (such as storage capacity constrain , spoliage of items , multi echlon inventory system)

makes “lowest cost sweet spot” extremely difficult to find.

Reinforcement Learning

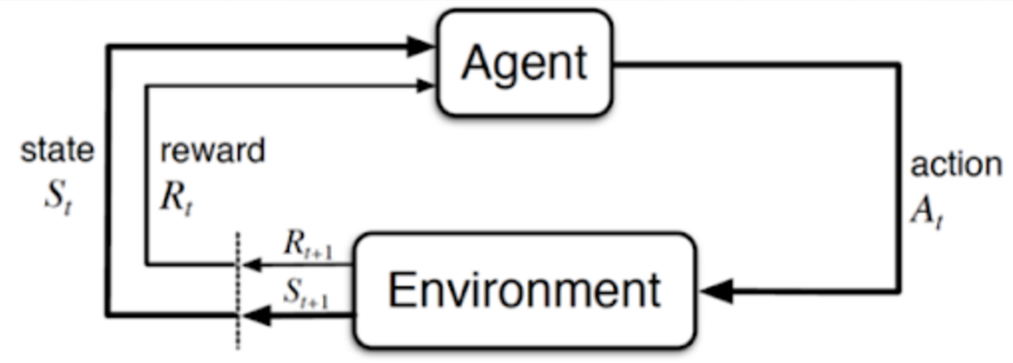
1. Trial-and-error learning through the interaction of a decision-maker called agent and a dynamic environment.
2. The environment gives rise to rewards
3. Agent tries to maximize the reward overtime learning near optimal policy
4. RL excels in **complex, uncertain environments that require sequential decision making.**



- The purpose of this project is to benchmark the agent's learned policy against already known optimal policy.

Markov Decision Process

1. RL ustilizes MDP framework
2. In MDP, there are four core elements ; States, Actions, Rewards and state transition probabilities
3. We need to identify what are states, actions , rewards, and state transition probabilites for our specific inventroy problem.



Problem description

Car Retail Store

At the start of each business day, Car Reatil Store checks its on-hand inventory of luxury cars. Customer demand is fulfilled immediately if stock is available, with daily sales following a normal distribution centered around 2 vehicles with standard deviation of 1 vehicle. At day's end, the inventory team reviews remaining stock and places orders to its manufacturer if needed, incurring a $5,000 fixed ordering fee per shipment. Ordered vehicles arrive precisely 2 days later. Holding costs of $50 per unsold vehicle per day accrue daily for storage and capital costs. When demand exceeds available stock, there is shortage cost of $1000. Shortage Cost includes both back order cost and cusotmer disatifaction cost. Over a 100-day planning horizon, the dealership aims to balance these costs—$50 daily holding per vehicle, $5,000 ordering fees, and $1,000 shortage cost — while navigating unpredictable demand fluctuations and fixed 2-day lead times.

(Data are generated)

Reinforcement Lerning Model

State space

Inventory position at the end of each day is considered as States.

Inventory position = on hand inventory + pending order - back log (back orders)

$S = \{0,1,2,3, \dots , 60\}$ (limited from 0 to 60 to reduce computational complexity)

Action Space

$A(s) = \{0,1,2,3, \dots , 30\}$ for all $s \in S$

Action 0 means No order is placed.

Reward = -(Holding Cost + Ordering Cost + Inventory Shortage Cost)

State Transition Probability

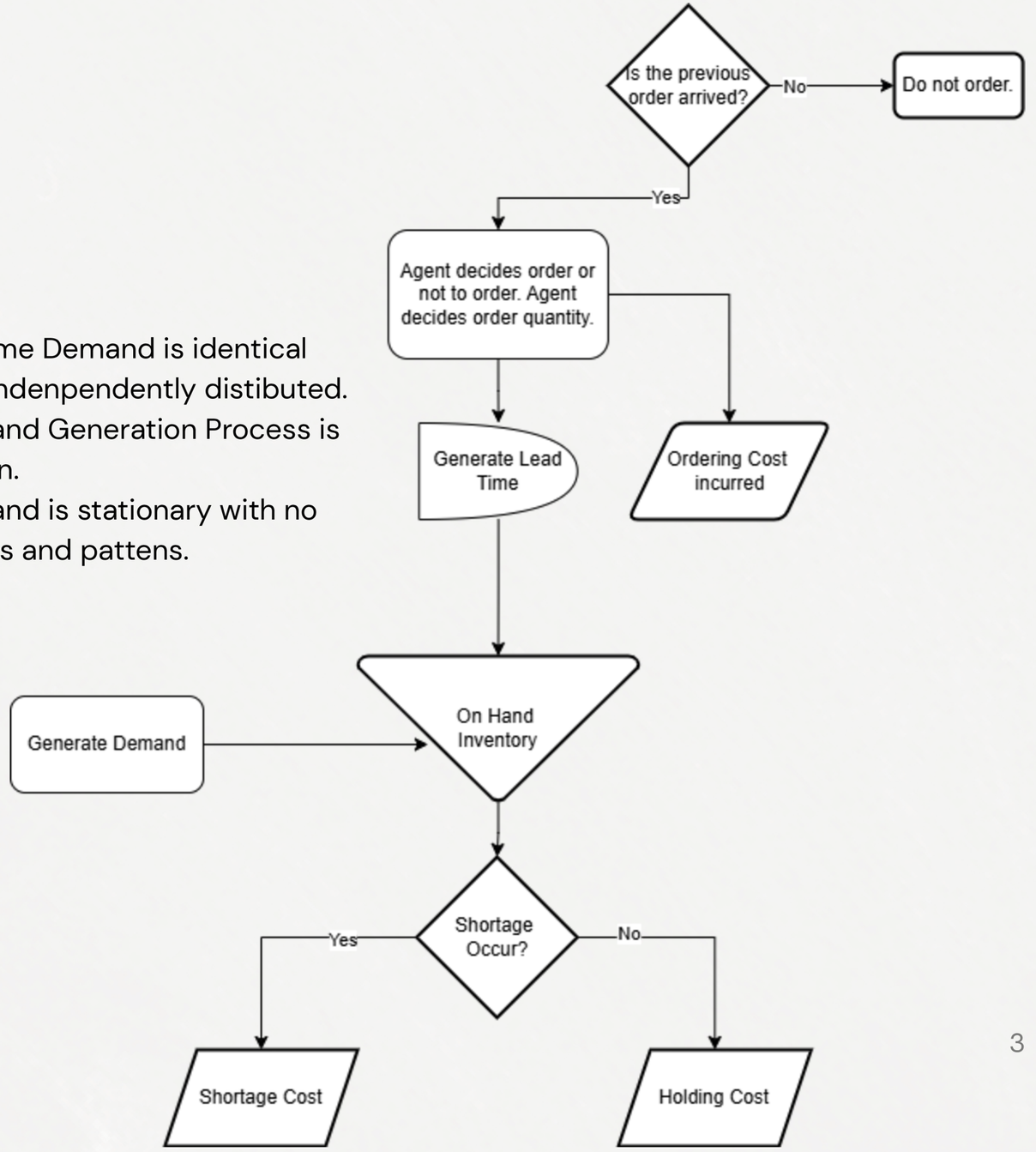
IRL methods are model free. Agent can automatically learn state transition probability through experience by interacting with environment simulation.

Simulation Model

Input

Daily Demand	N~ (2, 1) vehicle
Holding Cost / day	$50/vehicle
Ordering Cost	$5,000/order
Shortage Cost	$1,000/unmet demand
Planning horizon	100 days

- Assume Demand is identical and indenpendently distibuted.
- Demand Generation Process is known.
- Demand is stationary with no trends and pattens.



Output

- Total cost is passed to the agent as reward signal.

Q-learning

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

- It is off policy learning method ; soft policy (eg $\varepsilon\text{-}greedy\ policy$) is used as behavioral policy while greedy policy is used as target policy.

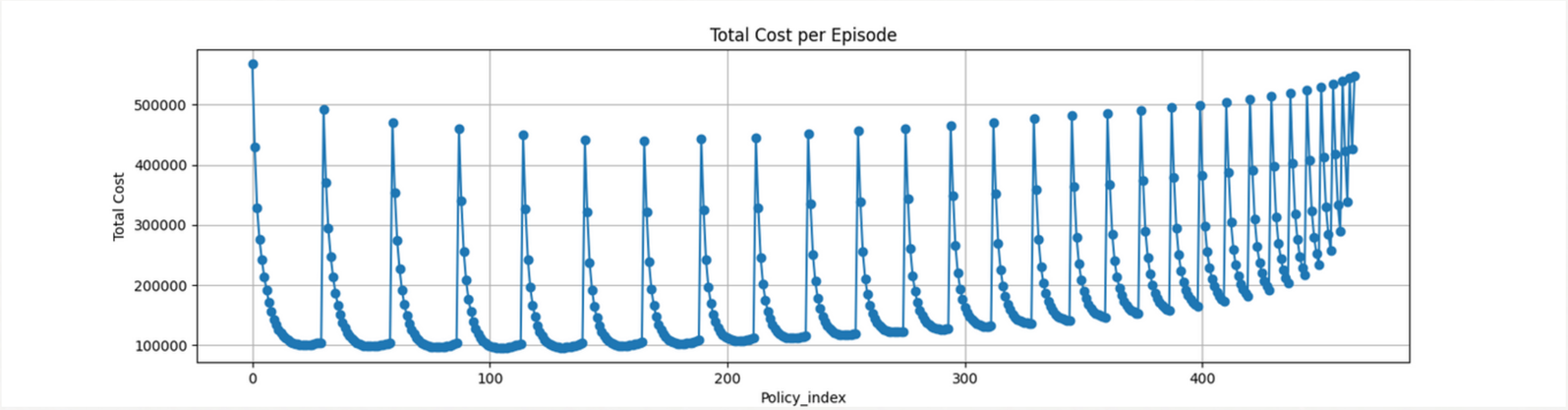
$\varepsilon\text{-}greedy\ policy$
An epsilon-greedy policy is a decision-making strategy that balances exploration and exploitation. The greedy action is performed most of the time (exploitation) while occasionally random action is selected (exploration) to discover potentially better options. Epsilon ,ε is probability that agent performs random action.

Base line policy to benchmark

Brute force search is used to find true global optimal policy. However, it is impossible to search every possible (s,S) since there is infinite possible (s,S) polices. Near optimal policy calculated using approximate EOQ method is (5,25) and true optimal policy should exist near this policy. To reduce search space, maximum S is set to 30.

Search space becomes,

(1,1) , (1,2) , (1,3) , … , (1,30)
(2,2) , (2,3) , …. , (2,30)
(3,3),(3,4) , …. , (3,30)
……
……
……
(28,29) , (28,30)
(30,30)



There are a total of 465 policies When maximum S is set to 30.

Optimal policy from the search turns out to be (4,21) with the mean total cost of $95287.
This mininum total cost is used as base line to benchmark Q-learning policy.

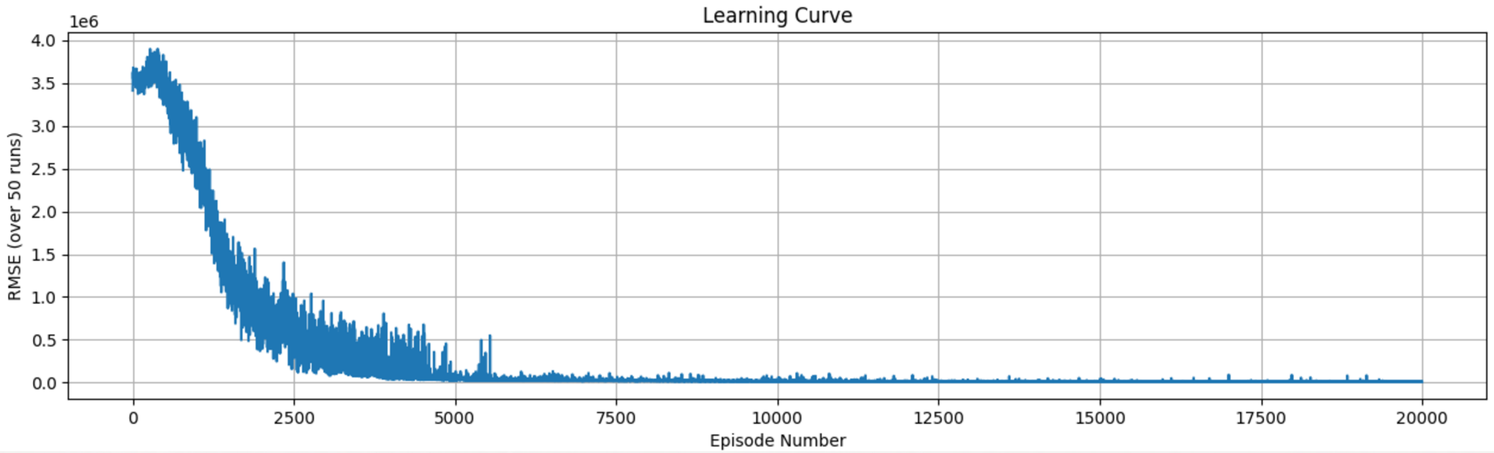
Training Q agent

- Q learning agent is trained with α=0.9, $epislon$ ε=0.9 with ε decay = 5 e-5, α decay = 5 e -5.
- Deacy equation

$$\alpha_t = \frac{\alpha_0}{1 + t \cdot d_\alpha} \quad \varepsilon_t = \frac{\varepsilon_0}{1 + t \cdot d_\varepsilon}$$

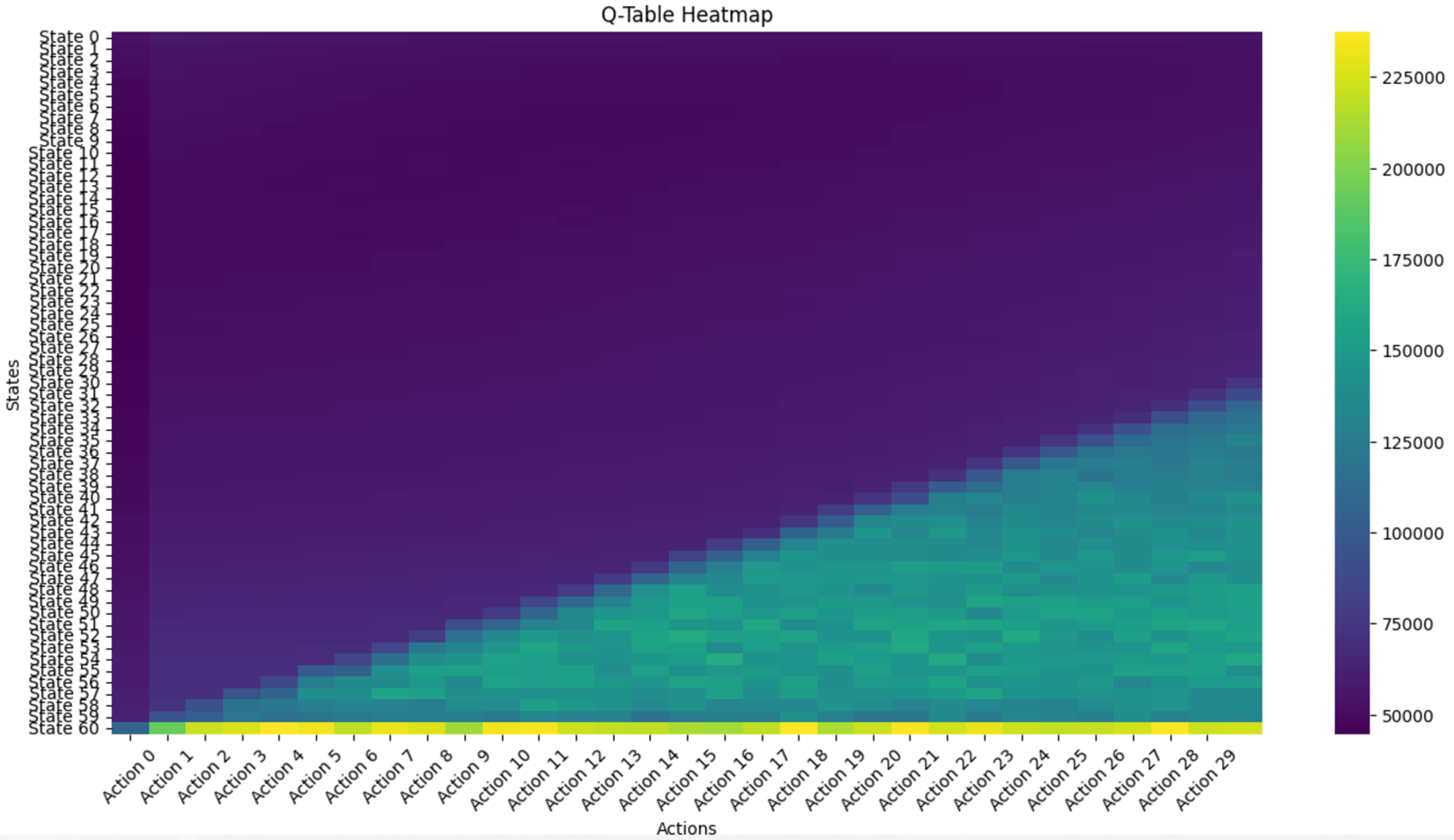
- Q learning algorithm is run for 20,000 episodes.
- Each episode has 100 days.
- 50 independent training round are run.
- Q values of 50 independent rounds are averaged.
- Avg Q table is used to get learned optimal policy.
- Learned optimal policy total cost is compared with searched global optimal (s,S) policy.

Result



- RMSE decreases gradually approaching 0.
- Q learning agent learns episodes by episodes getting closer to optimal policy.

Avg Q table


Q-Table Heatmap
States
Actions
Action 0
Action 1
Action 2
Action 3
Action 4
Action 5
Action 6
Action 7
Action 8
Action 9
Action 10
Action 11
Action 12
Action 13
Action 14
Action 15
Action 16
Action 17
Action 18
Action 19
Action 20
Action 21
Action 22
Action 23
Action 24
Action 25
Action 26
Action 27
Action 28
Action 29
225000
200000
175000
150000
125000
100000
75000
50000

Result

	Global Optimal	Q leaning
Mean Total Cost	95,287	96,259
Mean Shortage pribability	6 %	6.9%

total cost % error = 1%

- Q-learning achieves 99% of global optimal performance
- Slightly higher stockout risk but within practical tolerance

Conclusion

- Q-learning successfully learns near-optimal inventory policies for described problem, achieving 98% of global optimal cost.

Pros

- RL don’t need to fully understand environment dynamics, only need digital twin or simulation model
- RL can adapt to complex inventory system

Cons

- Computationally expensive : Requires extensive training episodes (30000+ episodes)
- converges to near-optimal (not global optimum) policies.
- Scabality issue: Tabular Q-learning struggles with high-dimensional states

Future Work

- Algorithm enhancement : Explore Deep RL (e.g., DQN) for scalability.
- Complex constraints : Model perishability/spoilage or storage limit with overstorage cost
- Non-stationary demand **:** Integrate time-varying trends/seasonality

The End

THANK YOU FOR LISTENING