Principles of Mathematical Economics applied to a Physical-Stores Retail Business

Ashwin Rao

ICME, Stanford University

March 15, 2019

Mathematical Economics and Retail Business

- Mathematical Economics is a vast and diverse area, consisting of different flavors of Optimization and Prediction problems
- We focus on a subset of these problems which are pertinent to running a real-world Retail Business
- In particular, many retail problems involve Stochastic Optimization
- ... that can be viewed from the abstract lens of Mathematical Econ.
- We can model these as Markov Decision Processes, eg:
    - How to *Supply* optimally given random demand and cost structures
    - How to *Price* optimally given random demand and supply
- Retail also involves forecasting problems, eg: Demand Forecasting
- ... as well as optimization problems on strategy/planning/scheduling

Inventory Control

- The fundamental problem in retail is Inventory Control
- How to move inventory optimally from suppliers to shoppers
- Let us view this from the lens of Mathematical Economics
- Abstracting to a Supply $\Rightarrow$ Demand optimization problem
- The two key foundations are the following simple problems:
    - Economic Order Quantity (EOQ) problem
    - Newsvendor Problem

Economic Order Quantity (EOQ)

- Demand for an item is a constant rate of μ units/year
- A new order is delivered in full when inventory reaches 0
- Fixed cost K for each order of non-zero units
- Holding cost h/unit/year for storage in store
- What is the optimal number of units to order?
- To minimize the annual cost of ordering + storage
- Note: Deterministic Demand is often an unreasonable assumption
- But EOQ is a useful foundation to build intuition
- Many extensions to EOQ (eg: EOQ for Perishables)
- EOQ concept goes beyond Retail (foundation in Mathematical Econ.)

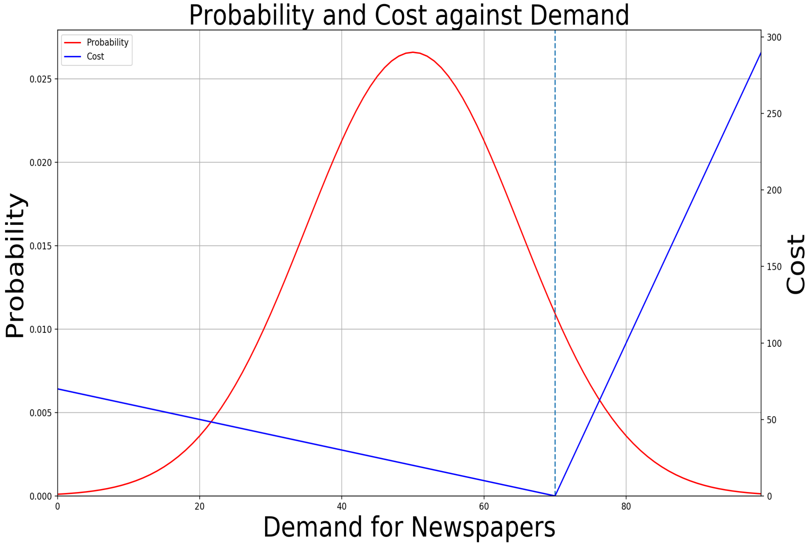
Solving EOQ

- Assume Q is the order quantity (Q^* is optimal order quantity)
- Then we order at annual frequency $\frac{\mu}{Q}$ (Period $\frac{Q}{\mu}$)
- Annual Ordering Cost is $\frac{\mu K}{Q}$
- Annual Holding Cost is $\frac{hQ}{2}$ (note: average inventory during year is $\frac{Q}{2}$)
- Annual Total Cost is:
$$\frac{\mu K}{Q} + \frac{hQ}{2}$$
- Taking derivative w.r.t. Q and setting it to 0 yields:
$$Q^* = \sqrt{\frac{2\mu K}{h}}$$

Newsvendor Problem

- Newsvendor problem is a single-period Inventory Control problem
- Daily demand for newspapers is a random variable x
- The newsvendor has an estimate of the PDF $f(x)$ of daily demand
- For each newspaper that stays unsold, we suffer a *Holding Cost h*
- Think of h as the purchase price minus salvage price
- For each newspaper we're short on, we suffer a *Stockout Cost p*
- Think of p as the missed profits (sale price minus purchase price)
- But p should also include potential loss of future customers
- What is the optimum # of newspapers to bring in the morning?
- To minimize the expected cost (function of f, h and p)

Newsvendor Problem



Solution to the Newsvendor problem

- For tractability, we assume newspapers are a continuous variable x
- Then, we need to solve for the optimal supply S that maximizes

$$g(S) = h \int_0^S (S - x) \cdot f(x) \cdot dx + p \int_S^\infty (x - S) \cdot f(x) \cdot dx$$

- Setting $g'(S) = 0$, we get:

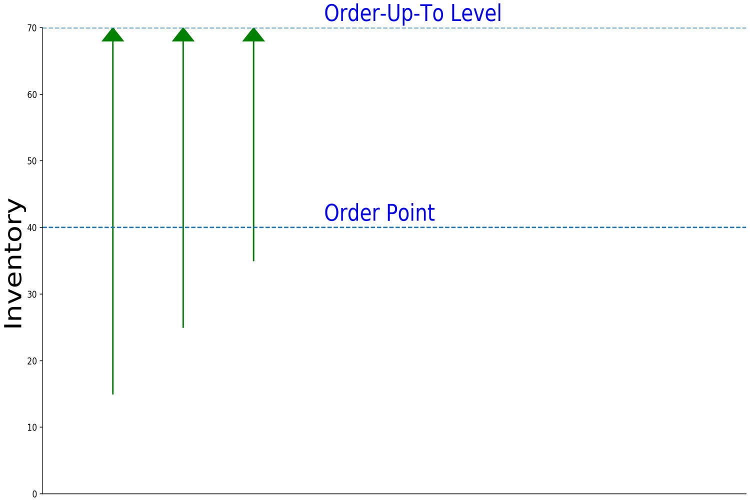
$$\text{Optimal Supply } S^* = F^{-1}(\frac{p}{p + h})$$

  where $F(y) = \int_0^y f(x) dx$ is the CDF of daily demand
- $\frac{p}{p+h}$ is known as the critical fractile
- It is the fraction of days when the newsvendor goes "out-of-stock"
- Assuming the newsvendor always brings this optimal supply S^*
- Solution details and connections with Financial Options Pricing here

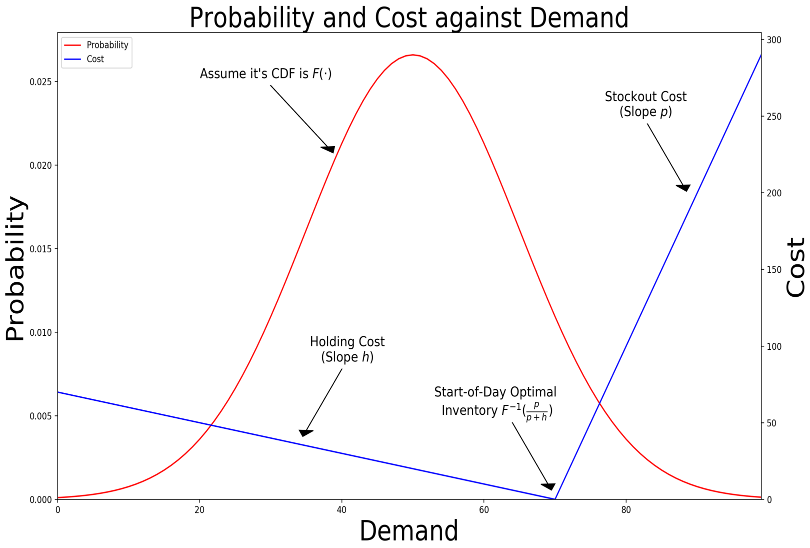
Multi-period: Single-store, Single-item Inventory Control

- The store experiences random daily demand given by PDF $f(x)$
- The store can order daily from a supplier carrying infinite inventory
- There's a cost associated with ordering, and order arrives in L days
- Like newsvendor, there's a Holding Cost h and Stockout Cost p
- This is an MDP where *State* is current Inventory Level at the store
- *State* also includes current in-transit inventory (from supplier)
- *Action* is quantity to order in any given *State*
- *Reward* function has h, p (just like newsvendor), and ordering cost
- Transition probabilities are governed by demand distribution $f(x)$
- This has a closed-form solution, similar to newsvendor fomula

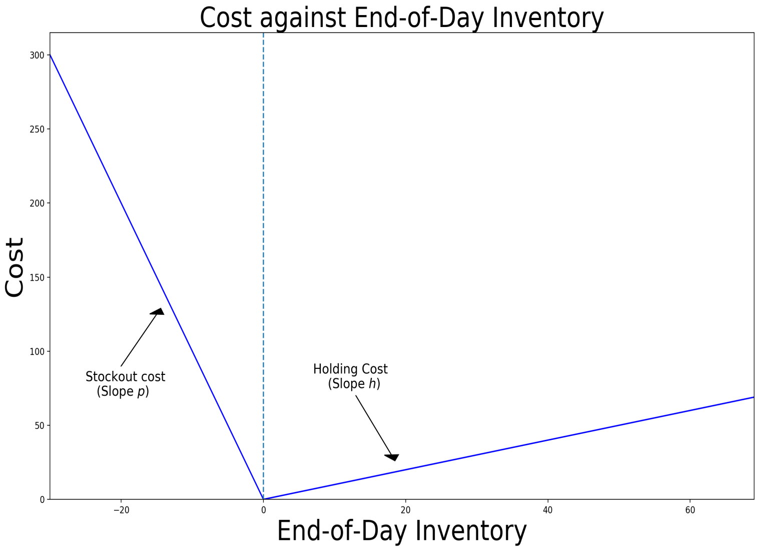
Optimal Ordering Policy (with Ordering Cost included)



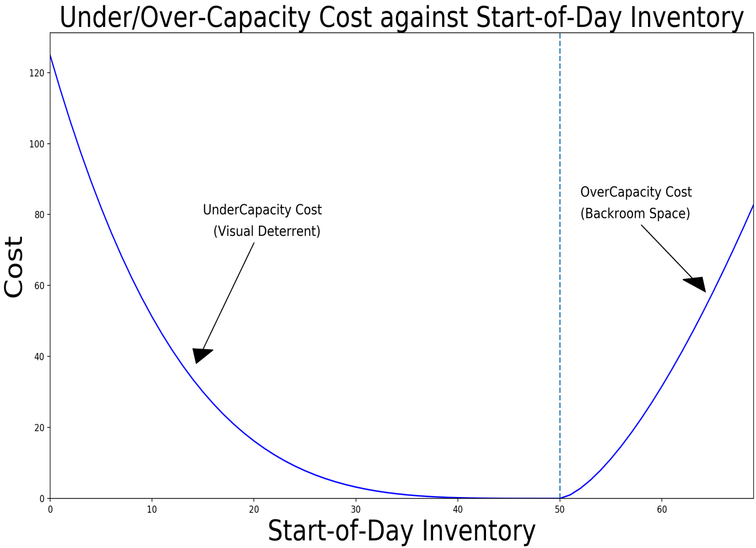
The Core of Textbook Problem has this Pictorial Intuition



Costs viewed against End-of-Day Inventory



UnderCapacity and OverCapacity Costs



UnderCapacity Cost: Customer Psychology and Economics

- Retail Mantra: "Stack it high and watch it fly"
- Customers like to see shelves well stocked
- Visual emptiness is known to be a sales deterrent
- So, full-looking shelves are part of presentation strategy
- At a certain level of emptiness, the deterrent rises sharply
- Hence the convex nature of this cost curve
- Note that this curve varies from item to item
- It also varies from regular season to end of season
- Modeling/calibrating this is tricky!
- However, getting a basic model in place is vital

OverCapacity Cost: Backroom Space Constraints

- Retail store backrooms have limited capacity
- Typically tens of thousands of items compete for this space
- Retailers like to have clean and organized backrooms
- A perfect model is when all your inventory is on store shelves
- With backroom used purely as a hub for home deliveries
- Practically, some overflow from shelves is unavoidable
- Hence, the convex nature of this curve
- Modeling this is hard because it's a multi-item cost/constraint
- Again, getting a basic model in place is vital

What other costs are involved?

- Holding Cost: Interest on Inventory, Superficial Damage, Maintenance
- Stockout Cost: Lost Sales, sometimes Lost Customers
- Labor Cost: Replenishment involves movement from truck to shelf
- Spoilage Cost: Food & Beverages can have acute perishability
- End-of-Season/Obsolescence Cost: Intersects with Clearance Pricing

Practical Inventory Control as a Markov Decision Process

- The store experiences random daily demand
- The store can place a replenishment order in casepack mutiples
- This is an MDP where *State* is current Inventory Level at the store
- *State* also includes current in-transit inventory (from warehouse)
- *Action* is the multiple of casepack to order (or not order)
- *Reward* function involves all of the costs we went over earlier
- State transitions governed by demand probability distribution
- Solve: Dynamic Programming or Reinforcement Learning Algorithms

Multi-node and Multi-item Inventory Control

- In practice, Inventory flows through a network of warehouses
- From source (suppliers) to destination (stores or homes)
- So, we have to solve a multi-"node" Inventory Control problem
- *State* is joint inventory across all nodes (and between nodes)
- *Action* is recommended movements of inventory between nodes
- *Reward* is the aggregate of daily costs across the network
- In addition, we have multi-item constraints
- Space and Throughput constraints are multi-item constraints
- So, real-world problem is multi-node and multi-item (giant MDP)

Clearance Pricing

- You are a few weeks away from end-of-season (eg: Christmas Trees)
- Assume you have too much inventory in your store
- What is the optimal sequence of price markdowns?
- Under (uncertain) demand responding to markdowns
- So as to maximize your total profit (sales revenue minus costs)
- Note: There is a non-trivial cost of performing a markdown
- If price markdowns are small, we end up with surplus at season-end
- Surplus often needs to be disposed at poor salvage price
- If price reductions are large, we run out of Christmas trees early
- "Stockout" cost is considered to be large during holiday season

MDP for Clearance Pricing

- *State* is [Days Left, Current Inventory, Current Price, Market Info]
- *Action* is Price Markdown
- *Reward* includes Sales revenue, markdown cost, stockout cost, salvage
- *Reward* & *State*-transitions governed by *Price Elasticity of Demand*
- Real-world *Model* can be quite complex (eg: competitor pricing)
- Ambitious Idea: Blend Inventory and Price Control into one MDP

Perspective from the Trenches (to solve real-world MDPs)

- I always start with a simple version of problem to develop intuition
- My first line of attack is DP customized to the problem structure
- RL Algorithms that are my personal favorites (links to lectures):
  - Deep Q-Network (DQN): Experience Replay, 2nd Target Network
  - Least Squares Policy Iteration (LSPI) - Batch Linear System
  - Exact Gradient Temporal-Difference (GTD)
  - Policy Gradient (esp. Natural Gradient, TRPO)
- Separate Model Estimation from Policy Optimization
- So we could customize RL algorithms to take advantage of:
  - Knowledge of transition probabilities
  - Knowledge of reward function
  - Any problem-specific structure that simplifies the algorithm
- Feature Engineering based on known closed-form approximations
- Many real-world, large-scale problems ultimately come down to suitable choices of DNN architectures and hyperparameter tuning

Inputs to these MDPs (other than the costs)

- Daily Demand Forecast probability distribution function
- Shelf Capacity
- Casepack size
- Lead Time (time from replenishment order to arrival on shelf)

Where do these inputs come from?

- From solutions to various other Forecasting and Planning problems
- Demand Forecasting is a statistical learning problem
- Planning problems are Optimization problems
- Some Planning Problems:
  - Assortment Selection
  - Shelf-size Planning
  - Casepack Sizing
  - Network Planning (for Lead Time)
  - Labor Planning
- Some planning problems need as input solution to Inventory Control
- How do we resolve this Chicken-and-egg situation?

The *Fixed-Point* of Planning and Control

- Planning problems are optimizations over parameter choices p
- For example, a set of Shelf-size choices or casepack choices
- Denote the Inventory Control MDP as D_p (p is input to MDP)
- Denote the Solution (Optimal Policy) to D_p as π_p^*
- Solve the planning problems (optimization) with input π_p^*
- The solution is the optimal parameter set p^*
- Feed p^* back into the MDP to solve for policy $\pi_{p^*}^*$
- Iterate this until we get stable p^* and $\pi_{p^*}^*$
- Very important to design the interfaces consistently
- Clean software framework for overall system design is vital to success