

# Stochastic Optimization Problems in Retail & Finance

Ashwin Rao

VP Data Science at Target & Adjunct Faculty at Stanford

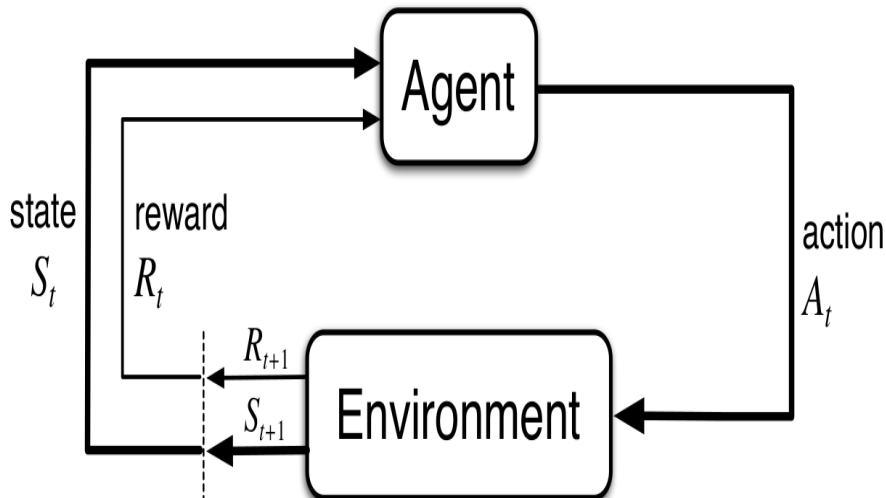
# My current world of Stochastic Optimization

- I am relatively new to the world of Stochastic Optimization
- Original background: Algorithms in Combinatorics & Abstract Algebra
- Wall Street work was in: Stochastic Calculus for Derivatives Pricing
- Joined Target 2.5 years ago to work on Supply-Chain Optimization
- Got introduced to Stochastic Optimization through Inventory Control
- Joined Stanford earlier this year, focusing on Mathematical Finance
- Next quarter, I am teaching a course on [RL for Financial Trading](#)
- I am also developing a [generic library](#) for Stochastic Optimization
- Exchange of ideas/techniques between Retail/Finance problems

# Overview

- 1 The Framework of Stochastic Optimization (MDP)
- 2 Core Problem in Retail: Inventory Control
- 3 Core Problem in Finance: Portfolio Optimization/Asset Allocation
- 4 Quick look at a few other problems in Retail and Finance
- 5 Finishing Comments

# The Stochastic Optimization Framework



# Components of the Framework

- The *Agent* and the *Environment* interact in a time-sequenced loop
- *Agent* responds to  $[State, Reward]$  by taking an *Action*
- *Environment* responds by producing next step's (random) *State*
- *Environment* also produces a (random) scalar denoted as *Reward*
- Goal of *Agent* is to maximize *Expected Sum* of all future *Rewards*
- By controlling the (*Policy* :  $State \rightarrow Action$ ) function
- This is a dynamic (time-sequenced control) system under uncertainty
- Formally known as a Markov Decision Process (MDP)

# Formal MDP Framework and Optimal Policy

- States  $S_t \in \mathcal{S}$  where  $\mathcal{S}$  is the State Space
- Actions  $A_t \in \mathcal{A}$  where  $\mathcal{A}$  is the Action Space
- Rewards  $R_t \in \mathbb{R}$  denoting numerical feedback
- Transitions  $p(s', r|s, a) = \Pr\{S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a\}$
- $\gamma \in [0, 1]$  is the Discount Factor for Reward when defining *Return*
- Return  $G_t = R_t + \gamma \cdot R_{t+1} + \gamma^2 \cdot R_{t+2} + \dots$
- Policy  $\pi(a|s)$  is probability that Agent takes action  $a$  in states  $s$
- Value function (under policy  $\pi$ )  $V_\pi(s) = \mathbb{E}[G_t | S_t = s]$  for all  $s \in \mathcal{S}$

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) \cdot (r + \gamma V_\pi(s')) \text{ for all } s \in \mathcal{S}$$

- Optimal Value Function  $V_*(s) = \max_\pi V_\pi(s)$  for all  $s \in \mathcal{S}$
- *There exists an Optimal Policy  $\pi_*$  achieving  $V_*(s)$  for all  $s \in \mathcal{S}$*

# Many real-world problems fit this MDP framework

- Self-driving vehicle (speed/steering to optimize safety/time)
- Game of Chess (Boolean *Reward* at end of game)
- Complex Logistical Operations (eg: movements in a Warehouse)
- Make a humanoid robot walk/run on difficult terrains
- Manage an investment portfolio
- Control a power station
- Optimal decisions during a football game
- Strategy to win an election (high-complexity MDP)

# Why are these problems hard?

- *State* space can be large or complex (involving many variables)
- Sometimes, *Action* space is also large or complex
- No direct feedback on “correct” *Actions* (only feedback is *Reward*)
- *Actions* can have delayed consequences (late *Rewards*)
- Time-sequenced complexity (eg: *Actions* influence future *Actions*)
- *Agent* often doesn't know the *Model* of the *Environment*
- “Model” refers to probabilities of state-transitions and rewards
- So, *Agent* has to learn the *Model* AND solve for the Optimal *Policy*



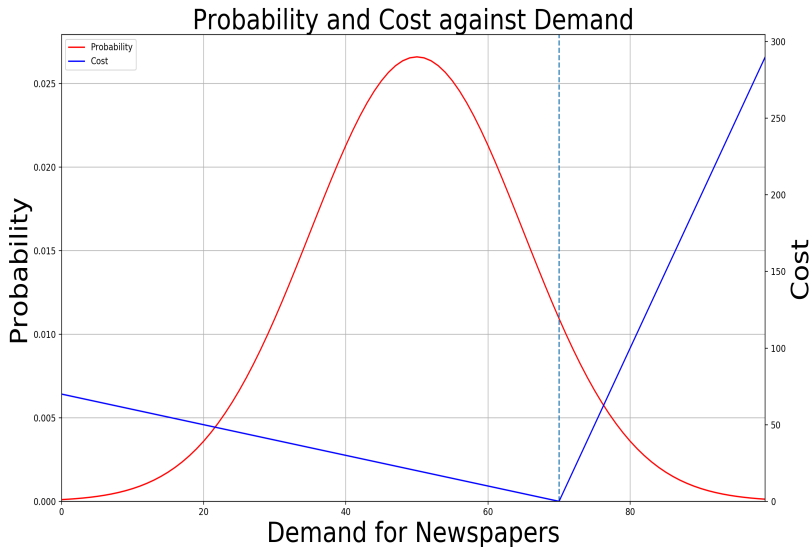
# Dynamic Programming versus Reinforcement Learning

- When Model is known, we do *Dynamic Programming* (DP)
- DP Algorithms exploit the recursive structure of Value function
- DP Algorithms do not require interactions with the environment
- But if Model is unknown, we need to do *Reinforcement Learning* (RL)
- RL Algorithms interact with the Environment in a loop, and learn
- Environment interaction could be real interaction or a simulator
- RL approach: Try different actions & learn what works, what doesn't
- RL Algorithms' key challenge is to tradeoff “explore” versus “exploit”
- DP or RL, Good approximation of Value Function is vital to success
- Deep Neural Networks are typically used for function approximation

# Inventory Control starts with the 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
- He incurs a “holding cost” of  $h$  per newspaper that stays unsold
- Think of  $h$  as the purchase price minus salvage price
- He incurs a “stockout cost” of  $p$  per newspaper that he is short on
- 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  $p$ )

# The 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:

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

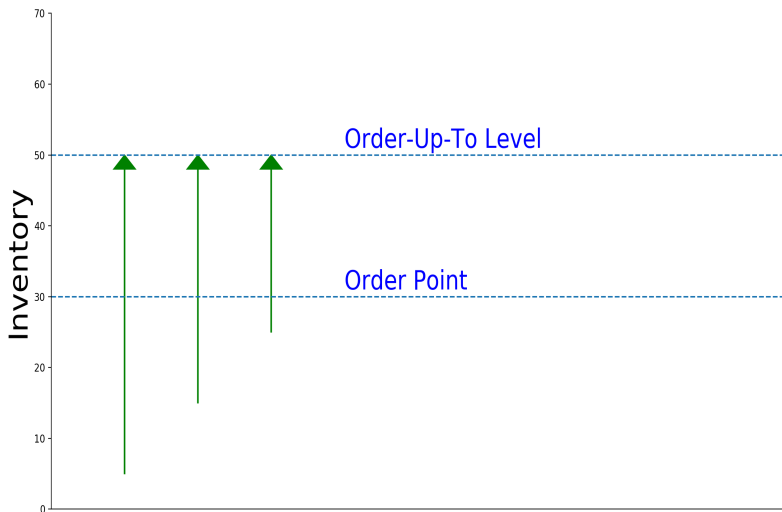
where  $F(y) = \int_0^y f(x)dx$  is the CDF of daily demand

- $\frac{p}{p+h}$  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 a 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 formula

# Optimal Policy: Order Point and Order-Up-To Level



# Adding real-world frictions and constraints

- Inventory is integer-valued, and orders are in casepack units
- Holding/stockouts costs are not linear functions of inventory
- Perishability, Obsolescence, End-of-season involve big costs
- Need to factor in labor costs of handling cases and singles
- Limits on shipping/receiving dates/times
- Often, there is a constraint on minimum presentation quantities
- Store inventory cannot exceed a threshold (eg: Shelf space)
- Supplier has constraints on min and max order quantities
- Uncertainty with the time for order arrival
- There are approximate closed-form solutions in some cases
- But general case requires generic DP or RL 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)



# Single-period Portfolio Optimization

- We start with a simple single-period portfolio optimization problem
- The simple setup helps develop understanding of the core concepts
- Assume we have one risky asset and one riskless asset
- The return (over the single-period) of the risky asset is  $\mathcal{N}(\mu, \sigma^2)$
- The return of the riskless asset is deterministic ( $= r < \mu$ )
- Start with \$1, aim to maximize our period-ending Expected Wealth
- This means we invest fully in the risky asset (since  $\mu > r$ )
- But people are risk-averse and will trade higher returns for lower risk
- The exact Risk-Return tradeoff is specified through *Utility of Wealth*
- *Utility* is a concave function of Wealth describing *Risk-Aversion*
- For an intro to Risk-Aversion and Utility Theory, [see here](#)
- The goal is to maximize period-ending **Expected Utility of Wealth**

# Solution to Single-period Portfolio Optimization

- $W$  denotes end-of-period wealth
- $\alpha$  denotes fraction invested in risky asset

$$W \sim \mathcal{N}(1 + r + \alpha(\mu - r), \alpha^2 \sigma^2)$$

- Let utility of wealth (concave) function be  $U(W) = -e^{-\gamma W}$
- Where  $\gamma$  is the coefficient (extent) of Risk-Aversion
- So we maximize over  $\alpha$ , the Expected Utility

$$\mathbb{E}[-e^{-\gamma W}] = -e^{-\gamma(1+r+\alpha(\mu-r)) + \frac{\gamma^2 \alpha^2 \sigma^2}{2}} = g(\alpha)$$

- Setting  $\frac{\partial \{\log g(\alpha)\}}{\partial \alpha} = 0$ , we get:

$$\alpha = \frac{\mu - r}{\gamma \sigma^2}$$

- This is an important investment fraction result
- Generalizes to multi-period and multiple risky assets

# Multi-Period: Merton's Portfolio Optimization Problem

- You will live for (deterministic)  $T$  more years
- Current Wealth + PV of Future Income (less Debt) is  $W_0 > 0$ .
- You can invest in (allocate to)  $n$  risky assets and a riskless asset
- Each asset has known normal distribution of returns
- Allowed to long or short any fractional quantities of assets
- Trading in continuous time  $0 \leq t < T$ , with no transaction costs
- You can consume any fractional amount of wealth at any time
- Dynamic Decision: Optimal Allocation and Consumption at each time
- To maximize lifetime-aggregated utility of consumption
- Consumption Utility assumed to have constant Relative Risk-Aversion

# Problem Notation

For ease of exposition, we state the problem for 1 risky asset

- Riskless asset:  $dR_t = r \cdot R_t \cdot dt$
- Risky asset:  $dS_t = \mu \cdot S_t \cdot dt + \sigma \cdot S_t \cdot dz_t$  (i.e. Geometric Brownian)
- $\mu > r > 0, \sigma > 0$  (for  $n$  assets, we work with a covariance matrix)
- Wealth at time  $t$  denoted by  $W_t > 0$
- Fraction of wealth allocated to risky asset denoted by  $\pi(t, W_t)$
- Fraction of wealth in riskless asset will then be  $1 - \pi(t, W_t)$
- Wealth consumption denoted by  $c(t, W_t) \geq 0$
- Utility of Consumption function  $U(x) = \frac{x^{1-\gamma}}{1-\gamma}$  for  $0 < \gamma \neq 1$
- Utility of Consumption function  $U(x) = \log(x)$  for  $\gamma = 1$
- $\gamma = (\text{constant})$  Relative Risk-Aversion  $\frac{-x \cdot U''(x)}{U'(x)}$

# Continuous-Time Stochastic Control Problem

- Balance constraint implies the following process for Wealth  $W_t$

$$dW_t = ((\pi_t \cdot (\mu - r) + r) \cdot W_t - c_t) \cdot dt + \pi_t \cdot \sigma \cdot W_t \cdot dz_t$$

- At any time  $t$ , determine optimal  $[\pi(t, W_t), c(t, W_t)]$  to maximize:

$$E\left[\int_t^T \frac{e^{-\rho(s-t)} \cdot c_s^{1-\gamma}}{1-\gamma} \cdot ds \mid W_t\right]$$

- where  $\rho \geq 0$  is the utility discount rate
- Think of this as a continuous-time Stochastic Control problem
- The *State* is  $(t, W_t)$
- The *Action* is  $[\pi_t, c_t]$
- The *Reward* per unit time is  $U(c_t)$
- The *Return* is the usual accumulated discounted *Reward*
- Find *Policy* :  $(t, W_t) \rightarrow [\pi_t, c_t]$  that maximizes the *Expected Return*
- Note:  $c_t \geq 0$ , but  $\pi_t$  is unconstrained

# Outline of Solution

- Optimal Value Function  $V^*(t, W_t)$  has a recursive formulation

$$V^*(t, W_t) = \max_{\pi, c} E[V^*(t_1, W_{t_1}) + \int_t^{t_1} \frac{e^{-\rho s} \cdot c_s^{1-\gamma}}{1-\gamma} \cdot ds]$$

- Then expressed as a Hamilton-Jacobi-Bellman (HJB) formulation

$$\max_{\pi_t, c_t} E[dV^*(t, W_t) + \frac{e^{-\rho t} \cdot c_t^{1-\gamma}}{1-\gamma} \cdot dt] = 0$$

- Standard HJB calculus (Ito's Lemma followed by partial derivatives w.r.t.  $\pi_t, c_t$ ) gives us a PDE for  $V^*(t, W_t)$
- We can reduce the PDE to an ODE with a guessed solution form
- The ODE has a standard solution
- The optimal allocation is given by:  $\pi^*(t, W_t) = \frac{\mu-r}{\sigma^2\gamma}$
- The optimal consumption is also a relatively simple closed-form
- All the gory details in [this lecture](#)

# Observations and Insights

- Optimal Allocation  $\pi^*(t, W_t)$  is constant (independent of  $t$  and  $W_t$ )
- Optimal Fractional Consumption  $\frac{c^*(t, W_t)}{W_t}$  depends only on  $t$
- Under Optimal Allocation, Expected Portfolio Return =  $\frac{(\mu-r)^2}{\sigma^2\gamma} + r$
- As  $T \rightarrow \infty$ , Optimal Fractional Consumption is a constant
- HJB Formulation was key and this solution approach provides a template for similar continuous-time stochastic control problems
- Analytical tractability was achieved due to assumptions of:
  - Normal distribution of asset returns
  - Constant Relative Risk-Aversion
  - Frictionless trading

# Porting this to real-world Portfolio Optimization

- Discrete Amounts of assets to hold and discrete quantities of trades
- Transaction costs
- Locked-out days for trading
- Non-stationary/arbitrary/correlated processes of multiple assets
- Changing/uncertain risk-free rate
- Consumption constraints
- Approximate Dynamic Programming or Reinforcement Learning
- Large Action Space points to Policy Gradient Algorithms



# Overview of a few other problems

- Financial Trading: American Options Pricing
- Financial Trading: Trade Order Execution
- Retail: Clearance Pricing

# American Options Pricing

- American option can be exercised anytime before option maturity
- Key decision at any time is to exercise or continue
- Backward Induction on a tree/grid is the default algorithm
- But it doesn't work for path-dependent options
- Also doesn't work when state dimension is large
- Longstaff-Schwartz's simulation-based algorithm is industry-standard
- RL is an attractive alternative to Longstaff-Schwartz
- RL is straightforward once Optimal Exercise is modeled as an MDP

# MDP for Optimal Options Exercise

- *State* is [Current Time, History of Underlying Security Prices]
- *Action* is Boolean: Exercise (i.e., Payoff and Stop) or Continue
- *Reward* always 0, except upon Exercise (= Payoff)
- *State*-transitions governed by Underlying Prices' Stochastic Process
- Optimal Policy  $\Rightarrow$  Optimal Stopping  $\Rightarrow$  Option Price
- All the details in [this lecture](#)
- Can be generalized to other Optimal Stopping problems

# Optimal Trade Order Execution (controlling Price Impact)

- You are tasked with selling a large qty of a (relatively less-liquid) stock
- You have a fixed horizon over which to complete the sale
- Goal is to maximize aggregate sales proceeds over horizon
- If you sell too fast, *Price Impact* will result in poor sales proceeds
- If you sell too slow, you risk running out of time
- We need to model temporary and permanent *Price Impacts*
- Objective should incorporate penalty for variance of sales proceeds
- Which is equivalent to maximizing aggregate Utility of sales proceeds

# MDP for Optimal Trade Order Execution

- *State* is [Time Remaining, Stock Remaining to be Sold, Market Info]
- *Action* is Quantity of Stock to Sell at current time
- *Reward* is Utility of Sales Proceeds (i.e., proceeds-variance-adjusted)
- *Reward & State-transitions* governed by *Price Impact Model*
- Real-world *Model* can be quite complex (Order Book Dynamics)

# Clearance Pricing

- You are a few weeks away from end-of-season (eg: Christmas Trees)
- Assume you have too much inventory in your store
- Also assume no more inventory replenishments are scheduled
- What is the optimal sequence of price reductions?
- So as to maximize our profit (sales revenue minus costs)
- If price reductions 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 peak season

# MDP for Clearance Pricing

- *State* is [Days Remaining, Current Inventory, Current Price]
- *Action* is Price Reduction
- *Reward* includes Sales revenue, stockout cost, salvage cost
- *Reward & State-transitions* governed by *Price Elasticity of Demand*
- Real-world *Model* can be quite complex (eg: competitor pricing)

# Finishing Comments

- Always start with a simple version of problem to develop intuition
- First line of attack should be DP customized to problem structure
- RL Algorithms that are my personal favorites:
  - 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
- Customize RL algorithms to take advantage of:
  - Knowledge of transition probabilities
  - Knowledge of reward function
  - Any other problem-specific features that simplify algorithm
- Engineer features based on known closed-form approximations
- Many real-world, large-scale problems ultimately come down to suitable choices of DNN architectures and hyperparameter tuning