Foundations of Reinforcement Learning with Applications in Finance

Ashwin Rao

Stanford University

A bit about me and about my book

- Co-Founder CXScore - AI to improve Customer Experience on Apps
- Adjunct Professor, Applied Mathematics (ICME), Stanford University
- Past: MD at Morgan Stanley, Trading Strategist at Goldman Sachs
- Wall Street career mostly in Rates and Mortgage Derivatives
- Educational background: Algorithms Theory and Abstract Algebra
- I direct Stanford's Mathematical & Computational Finance program
- Research & Teaching in: *RL and it's applications in Finance & Retail*
- Book: Foundations of RL with Applications in Finance
- Lived in Mumbai, LA, NYC, London, now settled in Palo Alto

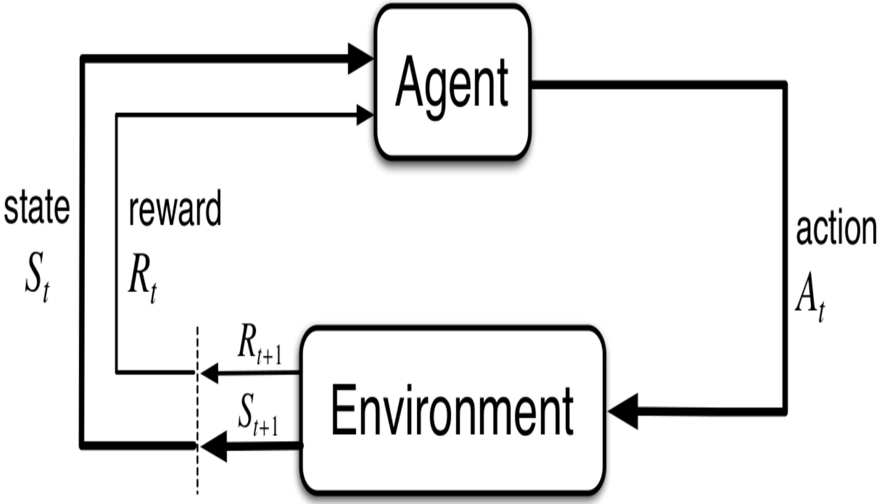
Key features of my book

- Book blends Theory, Modeling, Algorithms, Python, Trading problems
- Emphasis on broader principles in Applied Math & Software Design
- Focus on foundations and core understanding of concepts
- Tutorial-styled coverage, sometimes compromising rigor for intuition
- Significant emphasis on learning by coding the details
- 5 important financial applications covered in the book

AI for Dynamic Decisioning under Uncertainty

- Let's browse some terms used to characterize this branch of AI
- *Stochastic*: Uncertainty in key quantities, evolving over time
- *Optimization*: A well-defined metric to be maximized ("The Goal")
- *Dynamic*: Decisions need to be a function of the changing situations
- *Control*: Overpower uncertainty by persistent steering towards goal
- Jargon overload due to confluence of Control Theory, OR and AI
- For language clarity, let's just refer to this area as *Stochastic Control*
- The core framework is called *Markov Decision Processes* (MDP)
- *Reinforcement Learning* is a class of algorithms to solve MDPs

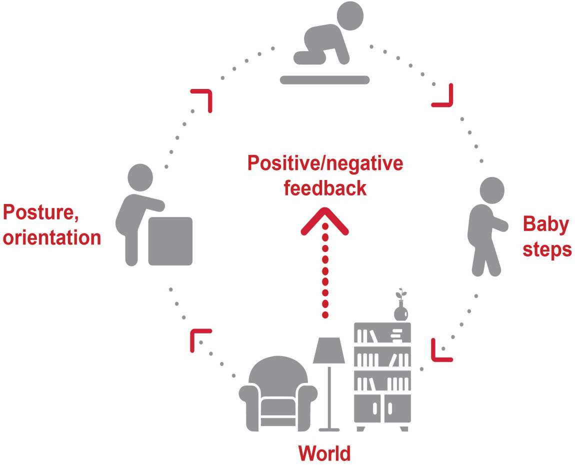
The MDP Framework



Components of the MDP 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*
- Each *State* is assumed to have the *Markov Property*
- Goal of *Agent* is to maximize *Expected Sum* of all future *Reward*s
- By controlling the ($Policy: State \rightarrow Action$) function
- This is a dynamic (time-sequenced control) system under uncertainty

How a baby learns to walk



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)

Self-Driving Vehicle

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*)
- Time-sequenced complexity (*Actions* influence future *States/Actions*)
- *Action*s can have delayed consequences (late *Reward*s)
- *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*
- *Agent Action*s need to tradeoff between "explore" and "exploit"

Dynamic Programming

- When Probabilities Model is known $\Rightarrow$ *Dynamic Programming* (DP)
- DP Algorithms take advantage of knowledge of probabilities
- So, DP Algorithms do not require interaction with the environment
- In the Language of AI, DP is a type of *Planning Algorithm*
- Why is DP not effective in practice?
    - Curse of Dimensionality
    - Curse of Modeling
- Curse of Dimensionality can be partially cured with Approximate DP
- To resolve both curses effectively, we need RL

Reinforcement Learning

- Typically in real-world, we don't have access to a Probabilities Model
- All we have is access to an environment serving individual transitions
- Even if MDP model is available, model updates can be challenging
- Often real-world models end up being too large or too complex
- Sometimes estimating a *sampling model* is much more feasible
- RL interacts with either *actual* or *simulated* environment
- Either way, we receive *individual transitions* to next state and reward
- RL is a "trial-and-error" approach linking *Actions* to *Rewards*
- Try different actions & learn what works, what doesn't
- This is hard because actions have overlapping reward sequences
- Also, sometimes Actions result in *delayed Rewards*

RL: Learning Value Function Approximation from Samples

- RL incrementally learns and improves from transitions data
- Deep Neural Networks used to estimate expected reward sums
- Big Picture: Sampling and Deep Learning estimates come together
- RL algorithms are clever about balancing "explore" versus "exploit"
- **Promise of modern A.I. is based on success of RL algorithms**
- Potential for automated decision-making in many industries
- In 10-20 years: Bots that act or behave more optimal than humans
- RL already solves various low-complexity real-world problems
- Possibilities in Finance are endless (book covers 5 key problems)

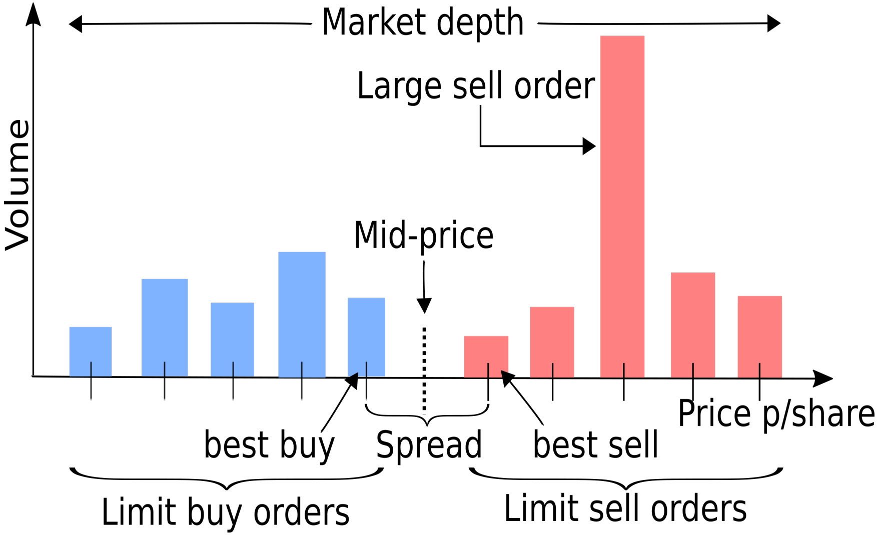
P1: Dynamic Asset-Allocation and Consumption

- The broad topic is Investment Management
- Applies to Corporations as well as Individuals
- The two considerations are:
  - How to allocate money across assets in one's investment portfolio
  - How much to consume for one's needs/operations/pleasures
- We consider the dynamic version of these dual considerations
- Asset-Allocation and Consumption decisions at each time step
- Asset-Allocation decisions typically deal with Risk-Reward tradeoffs
- Consumption decisions are about spending now or later
- Objective: Horizon-Aggregated Expected Utility of Consumption

P1: Consider the simple example of Personal Finance

- Broadly speaking, Personal Finance involves the following aspects:
  - Receiving Money: Salary, Bonus, Rental income, Asset Liquidation etc.
  - Consuming Money: Food, Clothes, Rent/Mortgage, Car, Vacations etc.
  - Investing Money: Savings account, Stocks, Real-estate, Gold etc.
- Goal: Maximize lifetime-aggregated Expected Utility of Consumption
- This can be modeled as a Markov Decision Process
- *State:* Age, Asset Holdings, Asset Valuation, Career situation etc.
- *Action:* Changes in Asset Holdings, Optional Consumption
- *Reward:* Utility of Consumption of Money
- *Model:* Career uncertainties, Asset market uncertainties

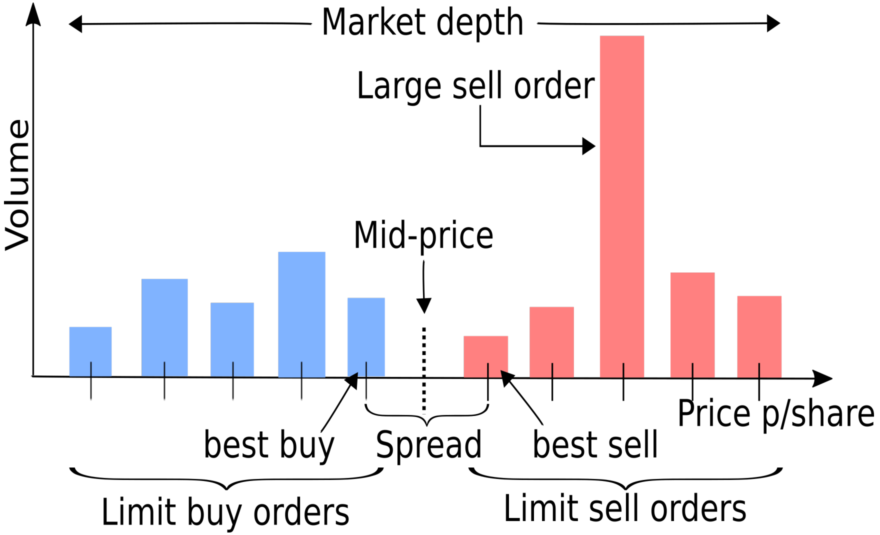
P2: Trading Order Book (abbrev. OB)



P2: Basics of Order Book (OB)

- Buyers/Sellers express their intent to trade by submitting bids/asks
- These are Limit Orders (LO) with a price P and size N
- Buy LO (P, N) states willingness to buy N shares at a price $\leq P$
- Sell LO (P, N) states willingness to sell N shares at a price $\geq P$
- Order Book aggregates order sizes for each unique price
- So we can represent with two sorted lists of (Price, Size) pairs
- We call best bid as simply *Bid*, best ask as *Ask*, their average as *Mid*
- We call *Ask* - *Bid* as *Spread*, *Worst Ask* - *Worst Bid* as *Market Depth*
- A Market Order (MO) states intent to buy/sell N shares at the *best possible price(s)* available on the OB at the time of MO submission

P2: Trading Order Book



P2: Price Impact and Order Book Dynamics

- A new Sell LO (P, N) potentially removes best bid prices on the OB
- After this removal, it adds to the asks side of the OB
- A new Buy LO operates analogously (on the other side of the OB)
- A Sell Market Order N will remove the best bid prices on the OB
- A Buy Market Order N will remove the best ask prices on the OB
- A large-sized MO can result in a big *Big-Ask Spread* - we call this the *Temporary Price Impact*
- *Spread* typically replenished by new LOs, potentially from either side
- Subsequent Replenishment moves *Bid/Ask/Mid* - we call this the *Permanent Price Impact*
- Price Impact Models with OB Dynamics can be quite complex

P2: Optimal Trade Order Execution Problem

- The task is to sell a large number N of shares
- We are allowed to trade in T discrete time steps
- We are only allowed to submit Market Orders
- Need to consider both *Temporary* and *Permanent* Price Impact
- For simplicity, consider a model of just the *Bid Price* Dynamics
- Goal is to maximize Expected Total Utility of Sales Proceeds
- By breaking N into appropriate chunks (timed appropriately)
- If we sell too fast, we are likely to get poor prices
- If we sell too slow, we risk running out of time
- Selling slowly also leads to more uncertain proceeds (lower Utility)
- This is a Dynamic Optimization problem
- We can model this problem as a Markov Decision Process (MDP)