Reinforcement Learning for Finance

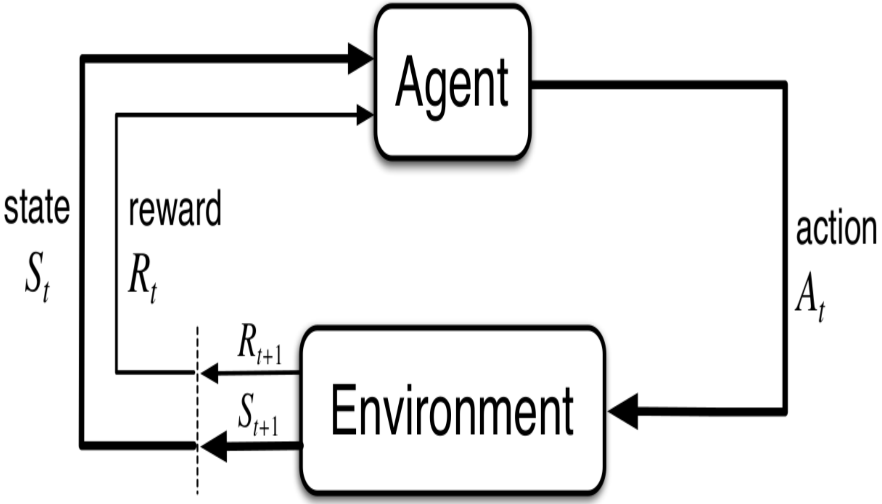
Ashwin Rao

Stanford University

A bit about me

- Adjunct Professor, Applied Math (ICME), Stanford University
- Previously spent 14 years at Goldman Sachs and Morgan Stanely
- Wall Street career mostly in Rates and Mortgage Derivatives
- I direct Stanford's Mathematical & Computational Finance program
- Research & Teaching in: *RL and it's applications to Finance & Retail*
- Book: Foundations of RL with Applications in Finance

The RL 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) *Reward*
- Goal of *Agent* is to maximize *Expected Sum* of all future *Reward*s
- This is a dynamic (time-sequenced control) system under uncertainty

Many real-world problems fit this 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 (highly complex)

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 experiences
- RL interacts with either *actual* or *simulated* environment
- RL is a "trial-and-error" approach linking *Actions* to *Returns*
- Try different actions & learn what works, what doesn't
- RL incrementally learns from a stream of data through interactions
- Big Picture: Sampling and Function Approximation come together
- Deep Neural Networks are typically used for function approximation
- **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 (I will cover 2 problems)

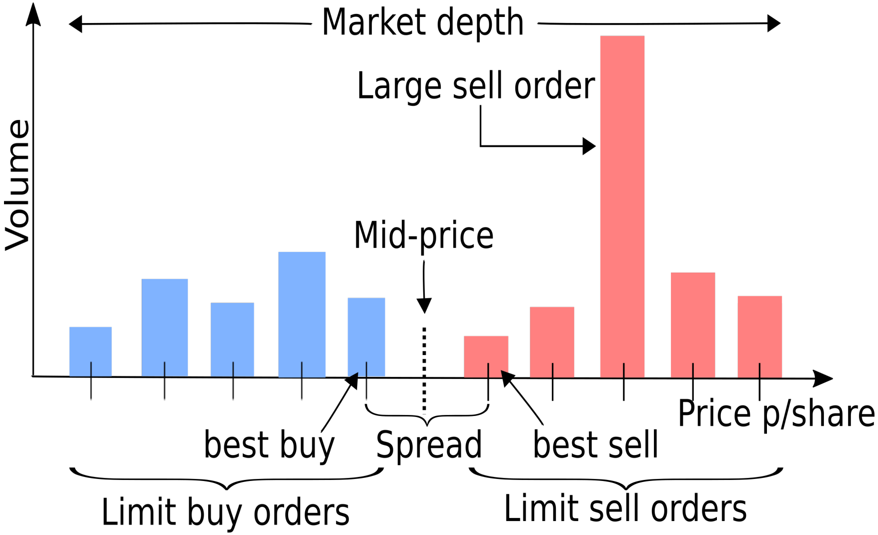
P1: Dynamic Asset-Allocation and Consumption

- 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
- We can model this in the RL framework as follows:
- *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: Order Book Dynamics and Price Impact

- 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 $\geq P$
- 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
- A Sell Market Order will remove the best bid 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 Stacks* - 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 block of shares in a finite amount of time
- We are only allowed to submit Market Orders
- Need to consider both *Temporary* and *Permanent* Price Impact
- Goal is to maximize Risk-Adjusted Total Sales Proceeds
- By breaking the block 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 leads to more uncertain proceeds ("risk-adjustment")
- This is a Dynamic Optimization problem

P2: Modeling in the RL framework

- *State* is current bid/ask stacks, and shares not yet sold
- *Action* is number of shares to be sold at this instant
- *Reward* is Utility of **price-impacted** sale proceeds
- Utility function does "risk-adjustment" of uncertain proceeds
- *Model* comprises of *Temporary* and *Permanent* Price Impact
- An RL algorithm will learn and optimize using a *simulated* OB
- The *simulated* OB is learnt by observing real OB dynamics