CME 241: Foundations of Reinforcement Learning with Applications in Finance

Ashwin Rao

ICME, Stanford University

Winter 2022
Meet your Instructor

- Joined Stanford ICME as Adjunct Professor in Fall 2018
- Research Interests: A.I. for Dynamic Decisioning under Uncertainty
- Technical mentor to ICME students, partnerships with industry
- Educational background: Algorithms Theory & Abstract Algebra
- 10 years at Goldman Sachs (NY) Rates/Mortgage Derivatives Trading
- 4 years at Morgan Stanley as Managing Director - Market Modeling
- Founded Tech Startup ZLemma, Acquired by hired.com in 2015
- One of our products was algorithmic jobs/career guidance for students
- Teaching experience: Pure & Applied Math, CompSci, Finance, Mgmt
Requirements and Setup

- **Pre-requisites:**
  - Undergraduate-level background in Applied Mathematics (Multivariate Analysis, Linear Algebra, Probability, Optimization)
  - Background in data structures/algorithms, fluency with numpy
  - Basic familiarity with Pricing, Portfolio Mgmt and Algo Trading, but we will do an overview of the requisite Finance/Economics
  - No background required in MDP, DP, RL (we will cover from scratch)

- Here’s [last year’s final exam](#) to get a sense of course difficulty

- Register for the course on Ed Discussion

- Install Python 3 and supporting IDE/tools (eg: PyCharm, Jupyter)

- Install LaTeX/Markdown and supporting editor for tech writing

- Download [the textbook for this course](#)

- Assignments and code in the textbook based on [this open-source code](#)

- *Fork* this repo and [get set up](#) to use this code in assignments

- Create separate directories for each assignment for CA ([Sven Lerner](#)) to review - send Sven your forked repo URL and *git push* often
Housekeeping

- Grade based on:
  - 25% 48-hour Mid-Term Exam (on Theory, Modeling, Programming)
  - 40% 48-hour Final Exam (on Theory, Modeling, Programming)
  - 30% Assignments: Technical Writing and Programming
  - 5% Participation: In Class, on Ed Discussion, during Office Hours


- First two weeks of lecture will be online (zoom link)

- Office Hours 12:30pm-2:30pm Fri (or by appointment) in ICME Mezzanine level, room M05 (within Huang Engg Bldg)

- You also have the option of joining these Office Hours on zoom

- Course Web Site: cme241.stanford.edu

- Ask Questions and engage in Discussions on Ed Discussion

- My e-mail: ashwin.rao@stanford.edu
Purpose and Grading of Assignments

- Assignments shouldn’t be treated as “tests” with right/wrong answer
- Rather, they should be treated as part of your learning experience
- You will truly understand ideas/models/algorithms only when you write down the Mathematics and the Code precisely
- Simply reading Math/Code gives you a false sense of understanding
- Take the initiative to make up your own assignments
- Especially on topics you feel you don’t quite understand
- Individual assignments won’t get a grade and there are no due dates
- The CA will review once every 2 weeks and provide feedback
- It will be graded less on correctness and completeness, and more on:
  - Coding and Technical Writing style that is clear and modular
  - Demonstration of curiosity and commitment to learning through the overall body of assignments work
  - Engagement in asking questions and seeking feedback for improvements
What is *Participation* and why does it matter?

- *Participation* means engagement and interactions
- With me, with the CA, and with other students
- In the classroom, or on Ed Discussion, or during Office Hours
- Come prepared to each class by reading the corresponding chapter
- Note down what you didn’t understand, and ask in class
- If it’s a deeper question, use Ed Discussion or Office Hours time
- The textbook is in manuscript version - provide feedback on typos and improvements by submitting issues in the RL-book git repo
- We want to bring back a vibrant culture of in-person interactions
- When you get a job, how you work with others matters A LOT!
- Teachers can teach best when students ask questions
Course based on my new **RL For Finance** book
Supplementary/Optional reading: **Sutton-Barto’s RL book**
I prepare slides for each lecture ("guided tour" of respective chapter)
A couple of lecture slides are from **David Silver’s RL course**
Code in my book based on **this open-source code**
Reading this code as important as the reading of the theory
We will go over some classical papers on the Finance applications
Some supplementary/optional papers from Finance/RL
All resources organized on the **course web site** ("source of truth")
Assignments: You can discuss solution approaches with other students
Because assignments are graded more for effort than correctness
Writing (answers/code) should be your own (don’t copy/paste)
You can invoke the core modules I have written (as instructed)

Exams: You cannot engage in any conversation with other students
Write to the CA if a question is unclear
Exams are graded on correctness and completeness
So don’t ask for help on how to solve exam questions
Open-internet Exams: Search for concepts, not answers to exam Qs
If you accidentally run into a strong hint/answer, state it honestly
Let’s browse some terms used to characterize this branch of A.I.

**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, O.R. and A.I.

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 \([\text{State}, \text{Reward}]\) 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*, meaning:
  - Next *State/Reward* depends only on Current *State* (for a given *Action*)
  - Current *State* captures all relevant information from *History*
  - Current *State* is a sufficient statistic of the future (for a given *Action*)
- 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
The following notation is for discrete time steps. Continuous-time formulation is analogous (often involving Stochastic Calculus):

- Time steps denoted as \( t = 1, 2, 3, \ldots \)
- Markov States \( S_t \in S \) where \( S \) is the State Space
- Actions \( A_t \in A \) where \( A \) is the Action Space
- Rewards \( R_t \in \mathbb{R} \) denoting numerical feedback
- Transitions \( p(r, s'|s, a) = \mathbb{P}[(R_{t+1} = r, S_{t+1} = s')|S_t = s, A_t = a] \)
- \( \gamma \in [0, 1] \) is the Discount Factor for Reward when defining Return
- Return \( G_t = R_{t+1} + \gamma \cdot R_{t+2} + \gamma^2 \cdot R_{t+3} + \ldots \)
- Policy \( \pi(a|s) \) is probability that Agent takes action \( a \) in states \( s \)
- The goal is find a policy that maximizes \( \mathbb{E}[G_t|S_t = s] \) for all \( s \in S \)
How a baby learns to walk

Positive/negative feedback

Posture, orientation

Baby steps

World
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*)
- Time-sequenced complexity (*Actions* influence future *States/Actions*)
- *Actions* can have delayed consequences (late *Rewards*)
- *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 Actions* need to tradeoff between “explore” and “exploit”
Value Function and Bellman Equations

- Value function (under policy \( \pi \)) \( V_\pi(s) = \mathbb{E}[G_t | S_t = s] \) for all \( s \in S \)

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

- Optimal Value Function \( V_*(s) = \max_\pi V_\pi(s) \) for all \( s \in S \)

\[
V_*(s) = \max_a \sum_{r,s'} p(r,s'|s,a) \cdot (r + \gamma V_*(s')) \quad \text{for all } s \in S
\]

There exists an Optimal Policy \( \pi_* \) achieving \( V_*(s) \) for all \( s \in S \)

- Determining \( V_\pi(s) \) known as Prediction, and \( V_*(s) \) known as Control
- The above recursive equations are called Bellman equations
- In continuous time, referred to as Hamilton-Jacobi-Bellman (HJB)
- The algorithms based on Bellman equations are broadly classified as:
  - Dynamic Programming
  - Reinforcement Learning
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*
- DP algorithms are iterative algorithms based on Fixed-Point Theorem
- Finding a *Fixed Point* of Operator based on Bellman Equation
- 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
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 Returns. 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 the Value Function from transitions data
- Appropriate Approximation of Value Function is key to success
- Deep Neural Networks are typically used for function approximation
- Big Picture: Sampling and Function Approximation come together
- RL algorithms are clever about balancing “explore” versus “exploit”
- Most RL Algorithms are founded on the Bellman Equations
- **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
- RL might soon be the most-desired skill in the technical job-market
- Possibilities in Finance are endless (we cover 5 important problems)
- Studying RL is a lot of fun! (interesting in theory as well as coding)
Many Faces of Reinforcement Learning

![Diagram showing the intersection of various fields including Computer Science, Engineering, Neuroscience, Mathematics, Psychology, Economics, Machine Learning, Optimal Control, Reinforcement Learning, Reward System, Operations Research, Classical/Operant Conditioning, Bounded Rationality.]

Ashwin Rao (Stanford)  "RL for Finance" course  Winter 2022  21 / 37
Vague (but in-vogue) Classification of Machine Learning

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Machine Learning
Overview of the Course

- Theory of Markov Decision Processes (MDPs)
- Dynamic Programming (DP) Algorithms
- Approximate DP and Backward Induction Algorithms
- Reinforcement Learning (RL) Algorithms
- Plenty of Python implementations of models and algorithms
- Apply these algorithms to 5 Financial/Trading problems:
  - (Dynamic) Asset-Allocation to maximize Utility of Consumption
  - Pricing and Hedging of Derivatives in an Incomplete Market
  - Optimal Exercise/Stopping of Path-dependent American Options
  - Optimal Trade Order Execution (managing Price Impact)
  - Optimal Market-Making (Bids and Asks managing Inventory Risk)
- By treating each of the problems as MDPs (i.e., Stochastic Control)
- We will go over classical/analytical solutions to these problems
- Then introduce real-world considerations, and tackle with RL (or DP)
- Course blends Theory/Math, Algorithms/Coding, Real-World Finance
Optimal Asset Allocation to Maximize Consumption Utility

- You can invest in (allocate wealth to) a collection of assets
- Investment horizon is a fixed length of time
- Each risky asset characterized by a probability distribution of returns
- Periodically, you are re-allocate your wealth to the various assets
- Transaction Costs & Constraints on trading hours/quantities/shorting
- Allowed to consume a fraction of your wealth at specific times
- Dynamic Decision: Time-Sequenced Allocation & Consumption
- To maximize horizon-aggregated *Risk-Adjusted Consumption*
- *Risk-Adjustment* involves a study of *Utility Theory*
State is [Current Time, Current Holdings, Current Prices]
Action is [Allocation Quantities, Consumption Quantity]
Actions limited by various real-world trading constraints
Reward is Utility of Consumption less Transaction Costs
State-transitions governed by risky asset movements
Derivatives Pricing and Hedging in an Incomplete Market

- Classical Pricing/Hedging Theory assumes “frictionless market”
- Technically, referred to as **arbitrage-free and complete market**
- *Complete market* means derivatives can be perfectly replicated
- But real world has transaction costs and trading constraints
- So real markets are incomplete where classical theory doesn’t fit
- How to price and hedge in an *Incomplete Market*?
- Maximize “risk-adjusted-return” of the derivative plus hedges
- Similar to Asset Allocation, this is a stochastic control problem
- Deep Reinforcement Learning helps solve when framed as an MDP
MDP for Pricing/Hedging in an Incomplete Market

- **State** is [Current Time, PnL, Hedge Qtys, Hedge Prices]
- **Action** is Units of Hedges to be traded at each time step
- **Reward** only at termination, equal to Utility of terminal PnL
- **State**-transitions governed by evolution of hedge prices
- Optimal Policy ⇒ Derivative Hedging Strategy
- Optimal Value Function ⇒ Derivative Price
An American option can be exercised anytime before option maturity
Key decision at any time is to exercise or continue
The default algorithm is Backward Induction on a tree/grid
But it doesn’t work for American options with complex payoffs
Also, it’s not feasible when state dimension is large
Industry-Standard: Longstaff-Schwartz’s simulation-based algorithm
RL is an attractive alternative to Longstaff-Schwartz
RL is straightforward once Optimal Exercise is modeled as an MDP
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
Can be generalized to other Optimal Stopping problems
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
Again, this amounts to maximizing Utility of sales proceeds
**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., Variance-adjusted-Proceeds)

**Reward & State-transitions governed by Price Impact Model**

Real-world Model can be quite complex (*Order Book Dynamics*)
Optimal Market-Making (controlling Inventory Buildup)

- Market-maker’s job is to submit bid and ask prices (and sizes)
- On the Trading *Order Book* (which moves due to other players)
- Market-maker needs to adjust bid/ask prices/sizes appropriately
- By anticipating the *Order Book Dynamics*
- Goal is to maximize *Utility of Gains* at the end of a suitable horizon
- If Buy/Sell LOs are too narrow, more frequent but small gains
- If Buy/Sell LOs are too wide, less frequent but large gains
- Market-maker also needs to manage potential unfavorable inventory (long or short) buildup and consequent unfavorable liquidation
- This is a classical stochastic control problem
MDP for Optimal Market-Making

- **State** is [Current Time, Mid-Price, PnL, Inventory of Stock Held]
- **Action** is Bid & Ask Prices & Sizes at each time step
- **Reward** is Utility of Gains at termination
- **State**-transitions governed by probabilities of hitting/lifting Bid/Ask
- Also governed by Order Book Dynamics (can be quite complex)
Week by Week (Tentative) Schedule

- W1: Markov Decision Processes
- W2: Bellman Equations & Dynamic Programming Algorithms
- W3: Backward Induction and Approximate DP Algorithms
- W4: Optimal Asset Allocation & Derivatives Pricing/Hedging
- W5: Options Exercise, Order Execution, Market-Making
- Mid-Term Exam
- W6: RL For Prediction (MC, TD, TD(\(\lambda\)))
- W7: RL for Control (SARSA, Q-Learning)
- W8: Batch Methods (DQN, LSTD/LSPI) and Gradient TD
- W9: Policy Gradient, Model-based RL, Explore v/s Exploit
- W10: Read-World RL and Guest Lecture by an Industry leader
- Final Exam
Some Landmark Papers we cover in this course

- Merton’s solution for Optimal Portfolio Allocation/Consumption
- Longstaff-Schwartz Algorithm for Pricing American Options
- Almgren-Chriss paper on Optimal Order Execution
- Avellaneda-Stoikov paper on Optimal Market-Making
- Original DQN paper and Nature DQN paper
- Lagoudakis-Parr paper on Least Squares Policy Iteration
- Sutton, McAllester, Singh, Mansour’s Policy Gradient Theorem
- Chang, Fu, Hu, Marcus’ AMS origins of Monte Carlo Tree Search
Similar Courses offered at Stanford

- AA 228/CS 238 (Mykel Kochenderfer)
- CS 234 (Emma Brunskill)
- CS 332 (Emma Brunskill)
- MS&E 338 (Ben Van Roy)
- EE 277 (Ben Van Roy)
- MS&E 251 (Edison Tse)
- MS&E 348 (Gerd Infanger)
- MS&E 351 (Ben Van Roy)
- MS&E 339 (Ben Van Roy)
Salient/Distinguishing features of this Course

- Emphasis on Foundations and Core Concepts
- More about *why* and *how*, versus *what*
- Balance between mathematical precision and intuitive understanding
- Coding from scratch, avoiding standard packages
- Encourages *Creator/Builder* mindset, versus *User* mindset
- Emphasis on code design driven by mathematical concepts/structures
- Key purpose of coding: Enables long-term retention of key learnings
- Several financial trading applications (and a couple from Retail)
- Coverage of continuous-time versions (elegant, analytical)
- I will dispel some common myths about industry versus academia