CME 241: Reinforcement Learning for Stochastic Control Problems in Finance

Ashwin Rao

ICME, Stanford University

Winter 2020
Meet your Instructor

- My 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
- I’ve been teaching short/medium-length courses for 25 years
- Topics across Pure & Applied Math, CS, Programming, Finance
- Current Interest: A.I. for Dynamic Decisioning under Uncertainty
- Current Industry Job: V.P. of A.I. at Target
- Joined Stanford ICME as Adjunct in Fall 2018
- Apart from CME 241, I am a technical mentor to ICME students
Requirements and Setup

(Light) Pre-requisites:

- Undergraduate-level background in Applied Mathematics (Linear Algebra, Probability Theory, Optimization)
- Background in Data Structures & Algorithms, with programming experience in numpy/scipy
- 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 these topics from scratch)

- Register for the course on Piazza
- Install Python 3 and supporting IDE/tools (eg: PyCharm, Jupyter)
- Note: Python 2 doesn’t support Type Annotations
- Create git repo for this course (for assignments/sharing)
- Message me on Piazza with your git repo URL (for reviews/grading)
- Install LaTeX and supporting editor (eg: TeXShop)
Housekeeping

- Grade based on:
  - 25% Mid-Term Exam (on Theory, Modeling, Algorithms)
  - 40% Final Exam (on Theory, Modeling, Algorithms)
  - 35% Assignments: Programming, Technical Writing, Theory Problems

- Lectures Wed & Fri 4:30pm - 5:50pm, Jan 8 - March 13
- Classes in Bldg 380 (Sloan Mathematics Ctr) - Room 380w
- Office Hours 2-4pm Fri (or by appointment) in my office (ICME M05)
- Course Web Site: cme241.stanford.edu
- Ask Questions and engage in Discussions on Piazza
- 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
- Rather, the entire body of assignments work will be graded
- 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
I recommend *Sutton-Barto* as the companion book for this course
   - I won’t follow the structure of Sutton-Barto book
   - But I will follow his approach/treatment

I will follow the structure of *David Silver’s RL course*
   - I encourage you to augment my lectures with David’s lecture videos
   - Occasionally, I will veer away or speed up/slow down from this flow

We will do a bit more Theory & a lot more coding (relative to above)

You can freely use my **open-source code** for your coding work
   - I expect you to duplicate the functionality of above code in this course

We will go over some classical papers on the Finance applications

To understand in-depth the analytical solutions in simple settings

I will augment the above content with many of my own slides

All of this will be organized on the course web site (“source of truth”)
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)**
The MDP Framework

- **State:** $S_t$
- **Reward:** $R_t$
- **Action:** $A_t$
- **Next State:** $S_{t+1}$

Flow diagram showing the interaction between the agent and the environment in a Markov Decision Process (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 \((\text{Policy} : \text{State} \rightarrow \text{Action})\) 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 $A_t \in \mathcal{A}$ 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 $G_t = R_{t+1} + \gamma \cdot R_{t+2} + \gamma^2 \cdot R_{t+3} + \ldots$
- 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 $s \in \mathcal{S}$
How a baby learns to walk

- Posture, orientation
- Positive/negative feedback
- World
- Baby steps

Positive/negative feedback

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)
Self-Driving Vehicle

Ride quality, arrival time

Location, velocity, sensor data

Steering, acceleration, braking

Traffic, signals, road conditions, routes
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 \( \pi \))**
  \[
  V_\pi(s) = \mathbb{E}[G_t | S_t = s] \text{ for all } s \in S
  \]
  \[
  V_\pi(s) = \sum_a \pi(a | s) \sum_{s', r} p(s', r | s, a) \cdot (r + \gamma V_\pi(s')) \text{ for all } s \in S
  \]

- **Optimal Value Function**
  \[
  V_\ast(s) = \max_{\pi} V_\pi(s) \text{ for all } s \in S
  \]
  \[
  V_\ast(s) = \max_a \sum_{s', r} p(s', r | s, a) \cdot (r + \gamma V_\ast(s')) \text{ for all } s \in S
  \]

- **There exists an Optimal Policy** \( \pi_\ast \) achieving \( V_\ast(s) \) for all \( s \in S \)
- Determining \( V_\pi(s) \) known as *Prediction*, and \( V_\ast(s) \) 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 versus Reinforcement Learning

- When Probabilities Model is known ⇒ *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 A.I, DP is a type of *Planning Algorithm*
- When Probabilities Model unknown ⇒ *Reinforcement Learning* (RL)
- RL Algorithms interact with the Environment and incrementally learn
- Environment interaction could be *real* or *simulated* interaction
- 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
Why is RL interesting/useful to learn about?

- RL solves MDP problem when *Environment Probabilities* are unknown
- This is typical in real-world problems (complex/unknown probabilities)
- RL interacts with *Actual Environment* or with *Simulated Environment*
- **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)
- Learning RL is a lot of fun! (interesting in theory as well as coding)
Many Faces of Reinforcement Learning
Vague (but in-vogue) Classification of Machine Learning

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Machine Learning

Ashwin Rao (Stanford)  “RL for Finance” course  Winter 2020
Overview of the Course

- Theory of Markov Decision Processes (MDPs)
- Dynamic Programming (DP) 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
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 Utility of Consumption
- Utility function represents degree of risk-aversion
- So, we effectively maximize aggregate Risk-Adjusted Consumption
MDP for Optimal Asset Allocation problem

- **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
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

In an incomplete market, we need to “choose” a risk-neutral measure

This amounts to specifying a Utility function

Maximizing “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 $\Rightarrow$ Derivative Hedging Strategy
- Optimal Value Function $\Rightarrow$ 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 path-dependent options
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 ⇒ Optimal Stopping ⇒ 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
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., Variance-adjusted-Proceeds)
- **Reward & State-transitions** governed by *Price Impact Model*
- Real-world *Model* can be quite complex (Limit Order Book Dynamics)
Optimal Market-Making (controlling Inventory Buildup)

- Market-maker’s job is to submit bid and ask prices (and sizes)
- On the Limit Order Book (which moves due to other players)
- Market-maker needs to adjust bid/ask prices/sizes appropriately
- By anticipating the Limit 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 Limit Order Book Dynamics (can be quite complex)
Week by Week (Tentative) Schedule

- W1: Markov Decision Processes & Overview of Finance Problems
- W2: Bellman Equations & Dynamic Programming Algorithms
- W3: Optimal Asset Allocation & Derivatives Pricing/Hedging
- W4: American Options Exercise & Optimal Trade Order Execution
- W5: Optimal Market-Making, and Mid-Term Exam
- W6: Model-free Prediction (RL for Value Function Estimation)
- W7: Model-Free Control (RL for Optimal Value Function/Policy)
- W8: RL with Function Approximation (including Deep RL)
- W9: Batch Methods (DQN, LSTDQ/LSPI), and Gradient TD
- W10: Policy Gradient Algorithms and Explore v/s Exploit
- W11: Final Exam
Getting a sense of the style and content of the lectures

A sampling of lectures to browse through and get a sense ...

- Understanding Risk-Aversion through Utility Theory
- HJB Equation and Merton’s Portfolio Problem
- Derivatives Pricing and Hedging with Deep Reinforcement Learning
- Stochastic Control for Optimal Market-Making
- Policy Gradient Theorem and Compatible Approximation Theorem
- Value Function Geometry and Gradient TD
- Adapative Multistage Sampling Algorithm (Origins of MCTS)
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)
- MS&E 251 (Edison Tse)
- CS 332 (Emma Brunskill)
- MS&E 338 (Ben Van Roy)
- MS&E 348 (Gerd Infanger)
- MS&E 351 (Ben Van Roy)