CS 498 Reinforcement Learning

Nan Jiang
Logistics

All information about this course: http://nanjiang.cs.illinois.edu/cs498/

• campuswire (please **self enroll**!)
  • main platform for course-related communications
  • you can post to instructors only; please post before requesting an individual meeting
• Slides & schedule (updated every week)
• textbook links
• TA & office hours (TBD)
• course policies
What’s this course about?

- A 400-level special topics course on reinforcement learning
- Will have some (minimum) programming components, but still lean heavily towards the mathematical side
- Think of this mainly as a mathematical course; the programming component is to ensure that you have some exposure to (some of) the actual algorithms
- “I want to do cool RL applications” — this course does not teach you that (but hopefully will give you a good foundation for it)
- “I want to add another cool ML course to my CV” — this course is probably not going to be useful
- Intellectual curiosity is the only reason you should take this course
- For a deeper course that prepares you for research in RL theory, check out my CS 598
Prerequisites

- Maths
  - Linear algebra, probability & statistics, basic calculus
  - Optional: Markov chains, stochastic processes, numerical analysis
- Exposure to ML
  - e.g., CS 446 Machine Learning
- We will soon have a homework 0 that helps you recall some of the math tools needed
Coursework & Grading

- Some readings after/before class
- Homework (60%): ~5 assignments, with both programming & writing components
  - Will likely to be handled via compass—more on this later
- Exam (40%): Either final or a late mid exam
- For 4 credit students: an additional course project (reproducing part of an empirical or theory paper)
Introduction to MDPs and RL
Reinforcement Learning (RL)
Applications

[Levine et al’16] [Ng et al’03] [Mandel et al’16] [Tesauro et al’07] [Singh et al’02] [Lei et al’12] [Mnih et al’15] [Silver et al’16]
Greedy is suboptimal due to delayed effects

Bellman Equation

\[ V^*(d) = \min\{3 + V^*(g) , 2 + V^*(f)\} \]

Need long-term planning
Shortest Path

The network diagram shows a graph with nodes labeled $s_0, c, b, d, e, f, g$ and weighted edges connecting them. The weights on the edges are as follows:

- $s_0$ to $c$: 2
- $c$ to $b$: 2
- $b$ to $d$: 4
- $d$ to $g$: 3
- $d$ to $f$: 1
- $f$ to $g$: 1
- $c$ to $e$: 4
- $e$ to $f$: 1

The shortest path from $s_0$ to $g$ is $s_0$ to $c$ to $e$ to $f$ to $g$, with a total weight of 8.
Stochastic Shortest Path

Markov Decision Process (MDP)

State transition distribution
Bellman Equation

\[ V^*(c) = \min \{4 + 0.7 \times V^*(d) + 0.3 \times V^*(e), 2 + V^*(e) \} \]

Stochastic Shortest Path

Greedy is suboptimal due to delayed effects

Need long-term planning
(also known as temporal credit assignment in RL)
Stochastic Shortest Path
via trial-and-error
Stochastic Shortest Path via trial-and-error

Trajectory 1: $s_0 \rightarrow c \rightarrow d \rightarrow g$

Trajectory 2:
Stochastic Shortest Path via trial-and-error

Trajectory 1: \(s_0 \rightarrow c \rightarrow d \rightarrow g\)

Trajectory 2: \(s_0 \rightarrow c \rightarrow e \rightarrow f \rightarrow g\)

…
**Stochastic Shortest Path via trial-and-error**

Nontrivial! Need **exploration**

- Assume states & actions are visited uniformly
- As we have more data, the estimated distributions will be closer to true distributions, and the learned policy will be closer to optimal
- Sample complexity: the amount of data needed to guarantee that a certain level of near-optimality is achieved
Random exploration can be inefficient

visited in $2^{-H}$ fraction of all trajectories

Freeway (one of the Atari games)
Video game playing

reward $r_t = R(s_t, a_t)$

objective: maximize $\mathbb{E} \left[ \sum_{t=1}^{H} r_t \mid \pi \right]$
Video game playing

Need generalization

Value function approximation
Video game playing

Find $\theta$ s.t.

$$f(x; \theta) \approx r + \gamma \cdot E_{x'}[f(x'; \theta)]$$

Need generalization

Value function approximation

$$f(\cdot ; \theta) \approx V^*$$
Adaptive medical treatment

- State: diagnosis
- Action: treatment
- Reward: progress in recovery

Formulating a real problem in the RL framework is difficult, and defining your state, action, horizon, and reward can be tricky—will come back later in the course.
Summary of this Part: 3 core challenges of RL

Temporal credit assignment
• A sequence of actions led to success/failure: which action(s) to attribute the consequence to?

Exploration
• How to take actions to collect a dataset that provides a comprehensive description of the environment?

Generalization
• How do deal with (very) large state spaces?
A Machine Learning view of RL
Lecture 1: Introduction to Reinforcement Learning

About RL

Many Faces of Reinforcement Learning

Computer Science
Economics
Mathematics
Engineering
Neuroscience
Psychology
Machine Learning
Optimal Control
Reward System
Reinforcement Learning
Operations Research
Classical/Operant Conditioning
Bounded Rationality

slide credit: David Silver
Supervised Learning

Given \( \{(x^{(i)}, y^{(i)})\} \), learn \( f : x \mapsto y \)

- Online version: for round \( t = 1, 2, \ldots \), the learner
  - observes \( x^{(t)} \)
  - predicts \( \hat{y}^{(t)} \)
  - receives \( y^{(t)} \)
- Want to maximize # of correct predictions
- e.g., classifies if an image is about a dog, a cat, a plane, etc. (multi-class classification)
- Dataset is fixed for everyone
- “Full information setting”
- Core challenge: generalization
Contextual bandits

For round $t = 1, 2, \ldots$, the learner

- Given $x^{(t)}$, chooses from a set of actions $a^{(t)} \in A$
- Receives reward $r^{(t)} \sim R(x^{(t)}, a^{(t)})$ (i.e., can be random)
- Want to maximize total reward
- You generate your own dataset $\{(x^{(t)}, a^{(t)}, r^{(t)})\}$!
- e.g., for an image, the learner guesses a label, and is told whether correct or not (reward = 1 if correct and 0 otherwise). **Do not know the true label.**
- e.g., for an user, the website recommends a movie, and observes whether the user likes it or not. **Do not know what movies the user really want to see.**
- “Partial information setting”
Contextual Bandits (cont.)

- Simplification: no $x$, Multi-Armed Bandits (MAB)
- Bandit is a research area by itself. we will not do a lot of bandits but will introduce some concepts that are useful for general RL
For round $t = 1, 2, \ldots,$

- For time step $h=1, 2, \ldots, H$, the learner
  - Observes $x_h^{(t)}$
  - Chooses $a_h^{(t)}$
  - Receives $r_h^{(t)} \sim R(x_h^{(t)}, a_h^{(t)})$
  - Next $x_{h+1}^{(t)}$ is generated as a function of $x_h^{(t)}$ and $a_h^{(t)}$ (or sometimes, all previous $x$’s and $a$’s within round $t$)
- Bandits + “Delayed rewards/consequences”
- The protocol here is for episodic RL (each $t$ is an *episode*).
A few misconceptions about RL
Two types of scenarios in RL research

1. Solving a large **planning** problem using a **learning** approach
   - e.g., AlphaGo, video game playing, simulated robotics
   - Transition dynamics (Go rules) known, but too many states
   - Run the simulator to collect data
   - RL proved to be successful

2. Solving a **learning** problem
   - e.g., adaptive medical treatment
   - Transition dynamics unknown (and too many states)
   - Interact with the environment to collect data
   - Great potential for RL, but not realized yet!
Is RL the magical blackbox in machine learning?

• More basic frameworks in machine learning: supervised/unsupervised learning

• Contextual bandit is an intermediate framework between SL & RL

• General rule: more flexibility/generality = less tractability!
  • RL is (often much) more general/flexible than SL / bandits
  • So, RL should be your last resort when the problem cannot be handled by any simpler framework
  • Be cautious about reducing an SL problem to RL—some are legit but many fall into the trap of reducing a simple problem to a more difficult one